UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Superior d'Enginyeria de Manresa

Control remot per veu d'un robot Open Source

7 de juny de 2019

Memòria del projecte que presenta Antoni Sbert i Cañellas
sota la direcció de l'Eng. Albert Babí Oller
per assolir el grau d'Enginyer en Sistemes TIC.

A tota la gent que m'ha acompanyat durant aquests anys, a tots els amics que he fet i als companys que m'han donat suport. En especial a la meva família i les persones més properes a mi. Destacant a les persones que m'han demostrat que m'estimen i m'han donat més suport, des del primer any que vaig arribar a la universitat. Donar les gràcies al meu tutor Albert Babí Oller, que m'ha ajudat molt a portar endavant aquest projecte i a la paciència que ha tingut durant aquest.

Índex

Resum

En l'actualitat la interacció amb les màquines mitjançant el reconeixement de la parla està molt de moda. Aquesta manera d'interactuar amb les màquines es duu desenvolupant des de fa molts anys. Aquest treball es centrarà en implementar una interfície basada en el reconeixement de la parla per controlar un sistema, en aquest cas un robot.

En aquest treball es fa un repàs de l'evolució del reconeixement de la parla fins a l'actualitat, examinant els avenços i teories que han portat a fer que sigui possible dur-ho a terme, quins dispositius i sistemes hi ha en l'actualitat i les seves característiques.

Per una altra banda, s'estudia l'evolució dels viatges espacials, en especial dels vehicles d'exploració espacial, anomenats també astromòbils. S'explica l'interès de l'aparició de projectes com l'Open Source Rover(OSR) i les seves variants.

Després d'estudiar els sistemes que existeixen, s'ha escollit Julius pel reconeixement de la veu i el Sawppy Rover com a rèplica d'astromòbil. S'expliquen els requeriments que es volen assolir. L'anàlisi del sistema, les modificacions tant de Julius com del rover i la implementació de l'estructura final. Obtenint com a resultat una interfície de reconeixement de la veu que controla el robot. Finalment, es realitza un petit resum de com en un futur podria evolucionar aquest projecte i com es podria millorar per poder dur a terme millores tant en la interfície com en el robot o si es podria usar en altres projectes.

Abstract

Nowadays the interaction between machines with speech recognition is fashionable. This way of interaction with machines has been developed for many years. This thesis is focused on the implementation of an interface based on speech recognition to control a system, in this case, a robot.

This thesis reviews the history of speech recognition, checking out the progress and theories that had made it possible, the devices and systems that are available and their characteristics.

From another point of view, this thesis also studies the history of space trips. It focuses on space exploration vehicles, also known as rovers. This study also explains the interest of projects like the Open Source Rover (OSR) and variants.

After studying the existing systems, Julius speech recognition and the Sawppy Rover were chosen. The requirements of the projectes are explained and also the necessary modifications to make the system work. Also how the final structure of the project has been done. As a conclusion, a little summary of how this project can improve from the robot point of view, the interface and how can be placed in other projects.

Part I.

Memòria

1. Introducció

L'ésser humà sempre ha tengut curiositat pel desconegut, la qual cosa l'ha portat a fer avenços de molts tipus durant la seva història. En l'àmbit tecnològic ha avançat especialment ràpid. Un exemple d'aquests avenços durant els anys seixanta, van ser els viatges espacials que varen ser la gran tendència del moment i es varen produir grans avenços que van portar a l'home a trepitjar la Lluna i a construir tot tipus d'astromòbils.

En l'actualitat ens trobem en una societat digitalitzada on tothom es troba connectat a Internet i amb màquines amb les que interactuar. Ha agafat força la utilització d'assistents intel·ligents que permeten una interacció més natural amb les persones. Moltes de les empreses grans, com per exemple Google o Amazon, s'han sumat a crear els seus propis assistents i un *hardware* per poder arribar a qualsevol tipus d'usuari. Això ha portat que en l'actualitat es doni una importància més gran al reconeixement de la parla i l'aplicació de mètodes com la intel·ligència artificial i el *machine learning*. En aquest projecte s'intentarà desenvolupar un sistema que la interacció sigui mitjançant el reconeixement de la parla.

Per això, en aquest projecte del primer que es parla és de la història del reconeixement de la veu, el tipus de reconeixements que hi ha i les característiques d'aquests, passant per com funcionen i els sistemes que estan implementats per poder millor-lo. La segona part parla de la història dels viatges espacials i es centrarà en els rovers que han existit i les rèpliques que la gent ha dut a terme. Un cop introduïts s'explicarà en què consistirà l'implementació del projecte: què necessitem per dur-lo a terme, el sistema de reconeixement de la parla que s'utilitza, l'astromòbil a simular, la integració del sistema i com s'ha duit a terme la robustesa per a què el sistema tingui el menor error possible.

2. Estat de l'art

2.1. Reconeixement de veu

2.1.1. Història del reconeixement de la parla

La interacció entre humans i màquines sempre ha estat un camp molt important per al desenvolupament tecnològic. A fi d'aconseguir un sistema amb el qual pugui haver-hi una interfície intuïtiva per a aquesta comunicació, el reconeixement de la veu és un dels sistemes que s'adequaria a aquest. El reconeixement de la parla és objecte d'estudi des dels anys 50.[BR04].

Inici del reconeixement de la parla

Durant els anys 50 es va plantejar la idea de reproduir la parla a fi de tenir màquines que parlessin com les persones. Els primers experiments consistien en usar un gramòfon connectat a tubs d'orgues i fer una primera aproximació a una veu reconeixible. Més endavant es va aconseguir fer el primer sintetitzador de veu mitjançant filtres. El que s'havia descobert des de tonalitats(com per exemple, la tonalitat de la nota Do) a espectres va servir per aproximar els fonemes a una ona sonora com la que produeix un instrument musical. Això va ser possible gràcies a la teoria dels fonemes, que descriu com es realitza acústicament la parla.

El 1952, els Laboratoris Bell van crear un sistema de reconeixement de dígits aïllats que reconeixia una quantitat limitada de fonemes[BR04]. Durant aquests anys també el MIT i els laboratoris RCA van desenvolupar sistemes que reconeixien fins a 10 síl·labes.

Als anys 60, diversos laboratoris japonesos varen decidir fer *hardware* exclusiu i màquines específiques per dur a terme el reconeixement. A més d'això, Sakai i Doshita al *Radio Research Lab in Tokyo*(RRLT) varen desenvolupar el primer reconeixedor de fonemes que segmentava el senyal d'entrada. Aquest va suposar el sistema precursor al reconeixement continuat de la veu. Durant aquest període també es va proposar la idea d'un reconeixement sense escala de temps que permetia arribar a reconèixer 2 fonemes alhora[Nil13]. Vintsyuk a la Unió Soviètica, va proposar afegir programació dinàmica a aquests processos. L'últim avenç durant aquests anys va ser el *Linear Predictive Code*(LPC) per Atal i Itakura, que consistia en estimar el que es deia a partir de l'ona rebuda modelant el senyal d'àudio com una combinació lineal d'un nombre de mostres anteriors més un senyal d'error.

Comercialització dels sistemes

Durant aquest període es va proposar un reconeixement de patrons sobre l'LPC per Itakura, Rabiner i Levinson, entre d'altres. L'inici de la primera empresa i producte del reconeixement de veu, va ser el VIP-100 per l'empresa Threshold Technology Inc, el 1972. Aquest producte va ser influenciat per *Advanced Research Project Agency*(ARPA) del govern dels Estats Units i el seu programa d'estudi del reconeixement de veu SUR (*Speech Understanding Research*). La Universitat de Carnegie Mellon, també influenciada per ARPA, va crear Harpy, un sistema de reconeixement de la parla que era capaç de reconèixer fins a 1011 paraules. Aquest usava

un sistema de reconeixement basat en grafs. A més a més, durant aquests anys va començar el desenvolupament DARPA's SUR incloent CMU's Hearsey-II and BBN's HWIM, que va aportar un nou sistema sobre xarxa de descodificació usant normes estrictes de fonètica.[ICT]

IBM i AT&T Bell Laboratories van començar a implementar sistemes de reconeixement automàtics amb l'objectiu de comercialitzar-los. IBM va crear el sistema Tangora que era propi per a cada usuari i s'havia d'entrenar. El sistema es basava en seqüenciar el senyal i fer-ne la transcripció a partir de l'anàlisi probabilístic, mètode que s'anomena *n-grams* [JM18].

L'objectiu d'aquestes empreses era fer un sistema que es pogués usar a les indústries de telefonia per interactuar amb els clients, però només era útil per a certes condicions ja que s'havia d'entrenar per a cada usuari. Això es va resoldre una vegada formulada la teoria dels models ocults de Markov(HMM) 2.1.2.

Mètodes d'aprenentatge i descodificació

La combinació entre les xarxes neuronals i els HMM no va ser possible fins als anys 90, on el primer sistema en implementar això va ser Sphinx de CMU. La Universitat de Cambridge, influenciada per DARPA com gairebé totes les empreses i laboratoris que investigaven sobre la veu, va crear el *framework* que es va convertir en l'eina d'investigació del reconeixement de la veu més comuna, el qual es va anomenar *Hidden Markov Model Tool Kit*(HTK)[BR04].

Malgrat que tots aquests sistemes de reconeixement de la veu tenien les seves funcionalitats, cap d'ells acabava de lidiar amb les diverses variants dels diferents parlants d'una llengua. Tampoc oferia la possibilitat d'interactuar amb un parlant de forma directa sense entrenament previ.

L'empresa AT&T's va desenvolupar el *Voice Recognition Call Processing*(VRCP), un sistema capaç de fer reconeixements i interactuar amb els usuaris. Aquesta idea es va usar per al sistema Pegasus i Jupiter a la Universitat MIT i per AT&T amb How May I Help You(HMIHY). Pegasus i HMIHY eren sistemes de parla per a fer gestions dels espais aeris dels aeroports incorporant sistemes de confirmació i reconeixement continu de la veu.

Gràcies als mètodes de *machine learning* que s'han desenvolupat més recentment, la millora de les *Artificial Neural Network*(ANN), la utilització de grans bases de dades i altres sistemes com el *Mel Frequency Cepstral Coefficients*(MFCC), s'ha pogut adaptar el reconeixement a les variacions dialectals. Un dels sistemes que aprofita el mètode de *machine learning* és el Deep Speech de Mozilla (Secció 2.2).

El reconeixement de veu permet comercialitzar productes i aplicacions que ofereixen comunicació directa entre les màquines i les persones. Actualment, està agafant força amb la comercialització d'aparells com el Google Home o l'Amazon Echo, uns del primers dispositius que han sortit al mercat que fan d'assistents domèstics mitjançant el reconeixement de la veu.

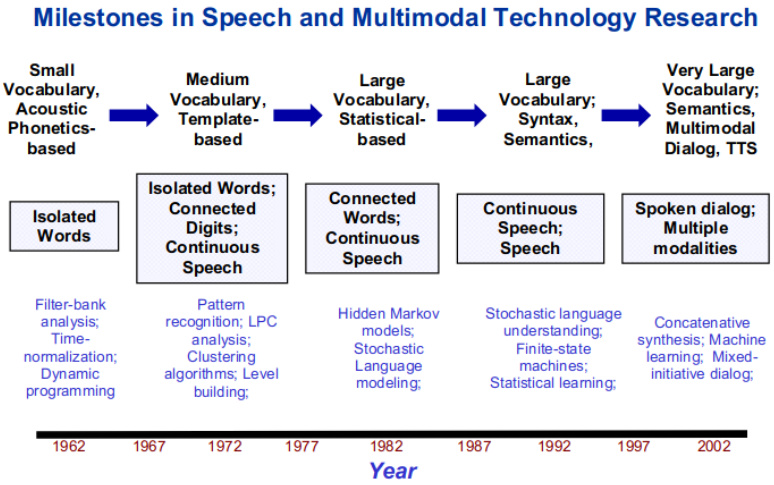

Figura 2.1.: Evolució del reconeixement de la veu durant els anys

Filtratge previ al reconeixement i models acústics i de llenguatge

Els sistemes de reconeixement de la parla més actuals tenen diferents mecanismes per poder diferenciar els factors que impedeixen fer un reconeixement objectiu(to, accent...), és a dir, permeten la transcripció literal de veu a paraules. Un d'aquests sistemes és el MFCC, que és un pas previ al reconeixement que permet ser usat per diversos usuaris.

Els models de més baix nivell usen sistemes que recullen des de fonemes fins a paraules. Aquests sistemes usen:

- Models acústics: recullen els fonemes i subfonemes en fitxers d'una llengua concreta, també existeixen els diccionaris fonètics, que agrupen els fonemes en paraules, la qual cosa permeten el seu reconeixement.
- Models de llenguatge: són fitxers que permeten ajustar la cerca a partir de les cerques anteriors, de forma similar al *machine learning* i permet optimitzar el sistema.

2.1.2. Hidden Markov Model

Les cadenes o processos de Markov simples són models estadístics que permeten fer prediccions d'esdeveniments a partir dels seus possibles estats i la probabilitat de transició. Una cadena de Markov necessita un conjunt d'estats finits($Estats = \{E_1, E_2, ..., E_n\}$), saber la probabilitat de passar d'un estat a un altre i que el temps de transició en diversos canvis d'estats sempre sigui el mateix.

Un exemple de problema d'un procés de Markov podria ser el clima. Dividirem el clima en 3 estats (bo, regular i dolent). Sabem que les probabilitats de transició entre estats són:

		Clima actual		
		Bo	Regular	Dolent
Pròxim dia	Bo	60%	30%	10%
	Regular	40%	30%	30%
	Dolent	50%	30%	20%

Taula 2.1.: Probabilitat de transició del clima

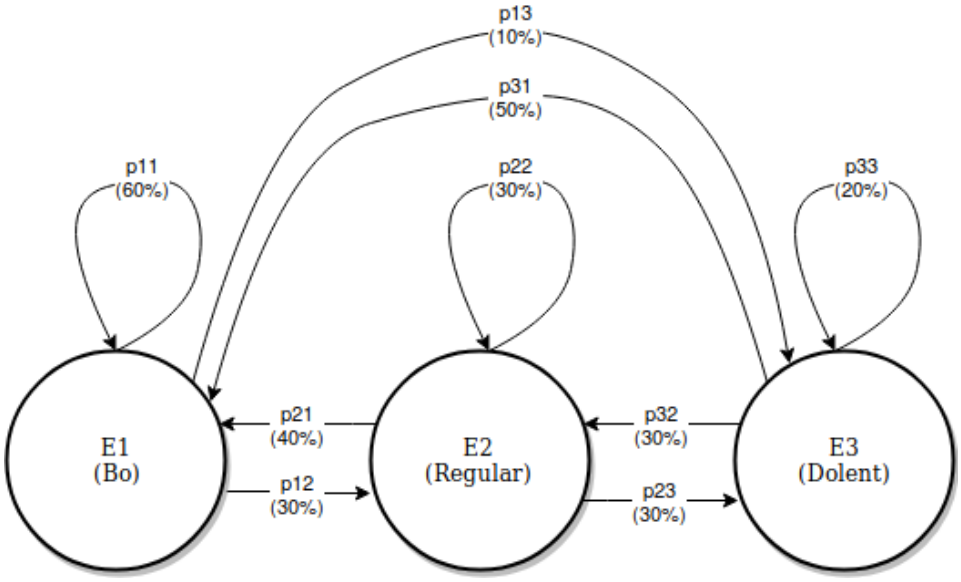

Figura 2.2.: Diagrama d'un model de Markov de 3 estats

Fins aquí, el problema és d'una cadena de Markov però, si afegim un petit matís, el problema es converteix en un sistema ocult de Markov (HMM). El HMM són cadenes de Markov [JJFG09] amb la diferència que no es pot veure l'estat directament. Els estats són ocults però tenen una sèrie d'accions determinades per a cadascun. La idea d'aquests models és predir l'estat resultant d'una acció a partir únicament de variables oferides pel sistema i l'estat anterior.

En el cas anterior podíem saber el temps que feia aquell dia però, ara no. Sabem que es mantenen les mateixes probabilitats que en el cas anterior de transició entre els estats. Per una altre banda, sabem que una persona duu a terme una sèrie d'activitats i només duu a terme una d'aquestes segons el temps que faci aquell dia. I sabem quina probabilitat té de dur a terme les diferents activitats segons el clima:

		Clima actual		
		Bo	Regular	Dolent
Accions	Correr	10%	20%	10%
	Netejar	10%	50%	10%
	Mirar TV	80%	30%	80%

Taula 2.2.: Probabilitat de dur a terme una acció segons el clima

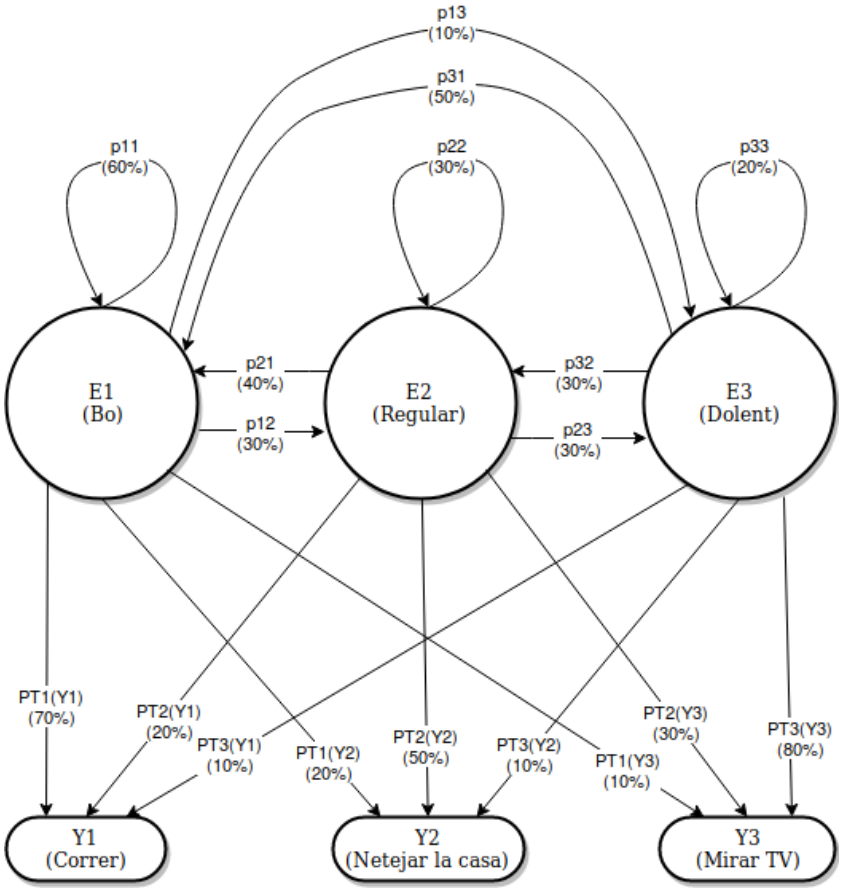

Figura 2.3.: Diagrama d'un model ocult de Markov de 3 estats

Per tant, sabem que ens trobem davant d'un models ocult de Markov, el terme d'ocult en aquests models es refereix al fet que no tenim coneixement directament de l'estat del clima només de les accions que fa aquesta persona i des d'aquí calcular la probabilitat de saber el temps que farà el pròxim dia o quin ha estat el temps aquell dia.

És possible usar aquests models en el reconeixement de la parla, ja que el senyal de la veu es pot dividir fàcilment. Normalment es segmenta en intervals de 10ms.

2.1.3. *Machine learning/artificial neural networks*

El mètode de *machine learning* forma part de la intel·ligència artificial. És un sistema amb el qual és possible fer que la màquina aprengui de forma autònoma i prengui decisions. Posem el cas d'un supermercat, que té els registres de totes les compres de cada persona. Aquestes dades sense processar no poden fer gran cosa, simplement coses simples com fer inventari. Ara bé, amb l'ajuda del *machine learning* es podrien arribar a treure patrons dels productes i ofertes amb més èxit i d'aquesta manera obtenir un avantatge competitiu respecte la competència. Això, aplicat al reconeixement de la parla, es pot aprofitar per a diversos patrons que hi ha en les diferents llengües.

Les xarxes neuronals artificials(ANN)[Bof94] són un sistema de software que es va crear amb l'objectiu de replicar les connexions neuronals humanes. Aquestes repliquen la sinapsi i permeten tenir un intercanvi d'informació i processament no lineal, que s'aprofita per a fer molts tipus de tasques. Aquestes estructures s'han d'entrenar amb exemples, per tant, la preparació i entrenament és lent.

Pel reconeixement, els mètodes de *machine learning* i ANN, són molt útils, permetent que la màquina dedueixi patrons de parla i poder ajustar les probabilitats. Per exemple, aquests mètodes serien capaços de deduir que després d'una pregunta, per parlar educadament, la següent paraula tendria unes altres probabilitats de ser: si us plau, per favor, gràcies... i ajustar les probabilitats en el sistema.

2.1.4. Filtratge del senyal d'entrada previ al reconeixement

Previament al reconeixement, el senyal d'àudio passa per un filtre que n'extreu la part subjectiva. Un d'aquest tipus de filtratge és l'MFCC[LME10]. Una vegada enregistrada el senyal de veu s'aplica sobre aquest senyal el MFCC. A grans trets consisteix a fer un processament del senyal d'aquesta forma:

1) Primer es segmenta el senyal en petits trams.

2) Després s'aplica la transformada discreta de Fourier i s'obté l'espectre del senyal; amb això és possible fer un filtratge

3) S'extreu la informació rellevant per a la comunicació objectiva, és a dir, extreure el to i altres aspectes subjectius de la parla.

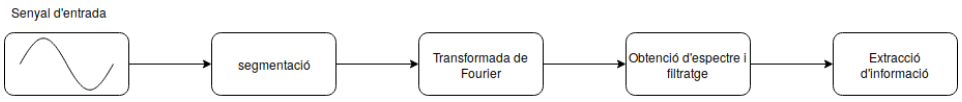

Figura 2.4.: Filtratge MFCC

2.1.5. Esquema general del reconeixement de veu

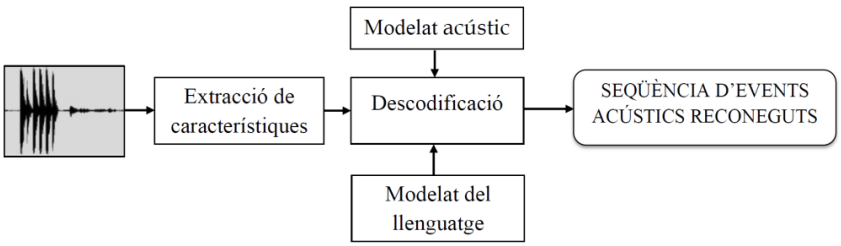

Figura 2.5.: Sistema de reconeixement de veu

Una vegada realitzat el filtrat, es fa el reconeixement. Malgrat que cada sistema té la seva manera específica de fer el reconeixement, en general segueixen l'estructura de la figura2.5. Comença amb una captació del senyal acústic Y i la divisió en un conjunt de petits vectors $Y = \{Y_1, Y_2, ..., Y_n\}$, el sistema intentarà fer un reconeixement d'aquests segments en paraules $\{W_1, W_2, ..., W_n\}$ que són la probabilitat de la seqüència que s'ha reconegut. Per tant, es pot calcular la màxima probabilitat de que aquest esdeveniment passi. Matemàticament pot ser expressat de la següent manera:

$$W = \arg\max_i P(W_i|Y)$$

Aplicant la fòrmula de Bayes:

$$W = \arg\max_i \frac{P(Y|W_i)P(W_i)}{P(Y)}$$

Sabent que les probabilitats de la observació P(Y) és independent a esdeveniments anteriors, podem prescindir de P(Y) per a cada esdeveniment.

$$W = \arg\max_i P(Y|W_i)P(W_i)$$

Per calcular la probabilitat d'aquestes seqüència de paraules s'utilitza un model acústic i un model de llenguatge.

Els models acústics permeten enregistrar aquesta probabilitat. Aquest models normalment estan fets a partir de HMM i de xarxes neuronals que milloren la seva aplicació. [Gel18].

Els models de llenguatge són usats com a complement dels models acústics per tenir en compte les limitacions lingüístiques i gramaticals d'una llengua. D'aquesta manera podem fer la predicció de la parla.[Blu14]

2.2. Software de reconeixement de veu

En un sistema de reconeixement de veu ens interessen saber diverses característiques del sistema per a poder dur a terme aplicacions i recerca sobre ells. Ens interessa saber quins sistemes són més aptes pel projecte:

- Si és un sistema Open Source, tenim accés al codi o privatiu, no tenim accés directament.

- Les plataformes i sistemes que es necessiten, per exemple, si només són útils per Linux o per algun altre.
- La velocitat del sistema, és a dir, el temps de resposta.
- Les especificacions que necessita el sistema com per exemple la memòria que ocupa en el sistema si es necessari 1GB de memòria o 1MB.
- Si es necessita d'un requeriment extra per a poder dur a terme reconeixements amb el sistema, com podria ser un sistema tancat que només funciona en un ser *hardware*.

Degut a la importància de requeriments extres en el reconeixement de la veu pel que fa al *hardware* extra dividim els sistemes en 2 grans grups.

1) Reconeixement directe sobre *hardware*:

   a) Julius: és un sistema de reconeixement de veu Open Source d'alt rendiment a temps real, de ràpid reconeixement pensat per desenvolupadors, usant un algoritme que es basa en N-gram i HMM que depèn del context, està pensat per ser usat a qualsevol mena de *hardware*. La plataforma principal es Linux però també és apte per Windows, Mac, Android i altres plataformes. Malgrat va ser desenvolupat per japonesos, també funciona en anglès.[git19] [org19a]

   b) Projecte de Mozilla (DeepSpeech + Common Voice): Project DeepSPeech[Moz19a], és un projecte Open Source fet per Mozilla, utilitzant tècniques de machine learning basdades en Badu's Deep Speech research. Aquest projecte per defecte utilitza la llibreria de Google Tensor flow[Goo18] que facilita la implementació. Ara bé, es dóna la possibilitat de canviar la llibreria i usar alguna altre compatible amb el projecte. És compatible amb MAC OS, Linux i Windows. Per una altra banda, hi ha Common Voice[Moz19b], una base de dades que recull Mozilla amb la finalitat de fer un model de reconeixement de veu prou bo.

   c) Simon: És un projecte Open Source a fi de substituir el ratolí i teclat, que usa *KDE libraries* (llibreria per fer aplicacions d'escriure sobre les aplicacions més simples), CMU Sphinx i Julius comentats anteriorment, que pot córrer tant en Windows com amb Linux.

   d) CMUSphinx: És un sistema de reconeixement Open Source proposat per la Universitat Carnegie Mellon que es pot usar tant en C, C++, C# com en Python. És un sistema de llibreries i eines que s'enllacen a aplicacions de reconeixement de veu, estan pensades tant per investigació com per ús comercial.[CMU18] [Shm18]

   e) Kaldi: Speech recognition Toolkit, dissenyat per ser usat en C++ que es pot usar tant per Linux com per Windows, va ser creat per la universitat Johns Hopkins amb la idea de crear un reconeixement de veu de baix cost i gran velocitat. La idea principal d'aquest projecte és que hi hagi un feedback de la comunitat que programa en C++. Ara bé l'únic problema és que és molt complex per gent que no té molt de coneixement de C++.[Res]

2) Reconeixement amb necessitat d'un *hardware* extra o un assistent:

a) Mycroft: És la versió de Linux dels famosos assistents de veu com per exemple, Siri d'Apple. Mycroft és un assistent Open Source que utilitza CMUSphinx per a fer el reconeixement de veu, concretament utilitza PocketSphinx. És Open Source i no està limitat a solament 1 tipus de *hardware*. Per això, té un software tant per Mark1 (el *hardware* que ven Mycroft), Raspberry Pi com per Android. Mycroft, és l'únic assistent que és possible de modificar a diferència de Siri, Google Assistant o Alexa.[Rit]

b) Siri: És l'assistent intel·ligent d'Apple, té un codi privatiu. Per poder usar aquesta tecnologia, és necessari un dispositiu de la marca Apple que porti preinstal·lat l'assistent. Ara bé, és possible arribar a comunicar la Raspberry Pi amb l'assistent mitjançant l'aplicació Hombridge.[Ram]

c) Google Assistant: És l'assistent intel·ligent de Google, té un codi privatiu. Ara bé, és possible accedir a crear aplicacions pròpies que es connecten a l'assistent i poder fer accions, en el nostre cas seria útil per poder connectar-nos amb la Raspberry Pi 3.[ins18]

d) Alexa: És l'assistent intel·ligent d'Amazon, té un codi privatiu.[ins] Ara bé, ens permet crear les anomenades "*skills*"que ens permeten connectar-nos amb els servidors d'Amazon i poder crear aplicacions que utilitzin el seu assistent.[pi17]

e) Google2Ubuntu: És un software que permet a Ubuntu fer reconeixement sobre algunes accions a Ubuntu.[elS16]

f) Software creat per tercers: La comunitat Open Source ha desenvolupat software de reconeixement de veu propi basat en llibreries.[Riv10]

2.3. Astromòbils

2.3.1. Història dels viatges espacials

Un dels grans temes on l'ésser humà sempre ha tingut interès és l'espai exterior. Durant el segle XX, l'ésser humà ha realitzat viatges cap a fora de la Terra. El 1942 el primer coet llançat per alemanys va començar una carrera espacial[Sci14a] entre els governs d'Alemanya i d'Estats Units. El míssil balístic, anomenat V-2[Bri14], va suposar una millora que marcaria els vehicles espacials, tant tripulats, com no tripulats.

La carrera armamentística entre alemanys i americans va durar fins el final de la Segona Guerra Mundial, quan la URSS agafaria el relleu dels alemanys.[Sci14d].

Durant molts anys les 2 grans potències varen fer diversos prototips i proves per poder portar vehicles a l'espai. Així, el 1957 es va llançar l'Sputnik 1[Mar09], el primer satèl·lit que orbitava la Terra. Aquest satèl·lit va estar orbitant durant 92 dies fins a la seva caiguda i destrucció en entrar en contacte amb l'atmosfera. Els Estats Units, en resposta a aquest esdeveniment, varen crear la NASA, una agència que s'encarregaria de programes espacials no militars.

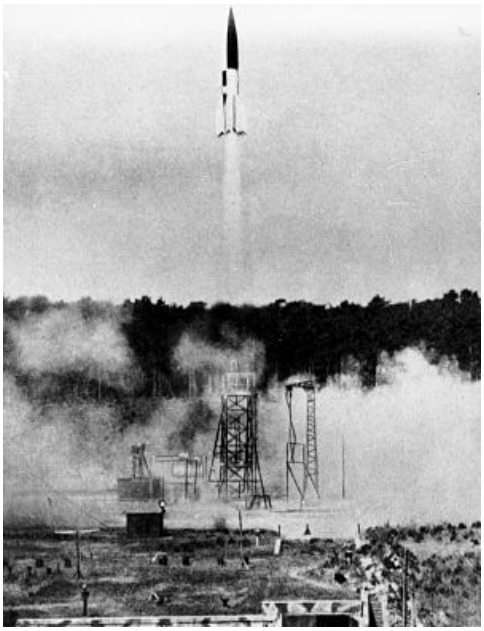

Figura 2.6.: Imatges del míssil balístic V2

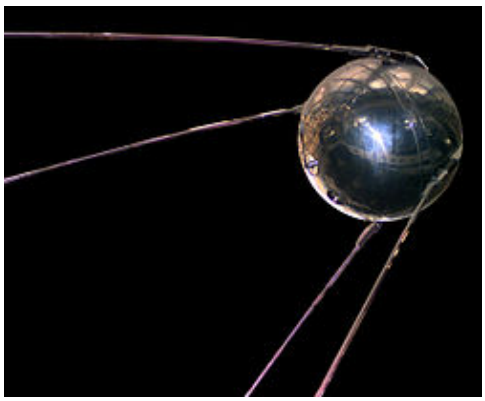

Figura 2.7.: Imatge del primer satèl·lit Sputnik 1

A partir d'aquí es varen començar a crear diverses naus per ambdues potències com l'Sputnik 2 (1954) amb el fi de poder enviar primer diversos animals com la famosa gossa Laika que anava a dins, i més endavant humans i també, augmentar les comunicacions amb el llançament de satèl·lits. El 1961 el rus Yuri Gagarin va arribar a l'espai convertint-se així en el primer ésser humà que anava a l'espai exterior. El 1969 mitjançant el programa Apollo de la NASA, va arribar el primer home a la Lluna. Tot seguit, es varen fer diverses missions cap a la Lluna. Però després d'aquestes va començar a decaure el programa lunar.

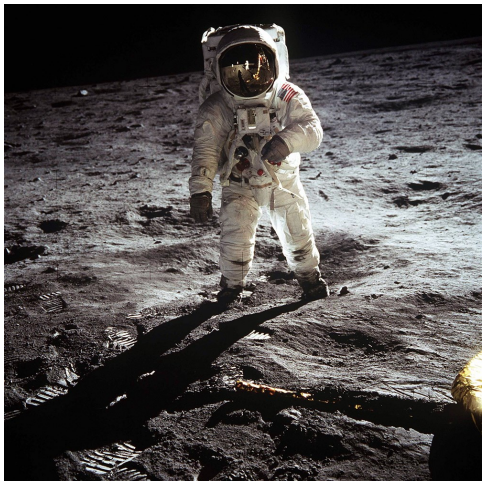

Figura 2.8.: Imatge de la missió Apollo 11

2.3.2. Història dels astromòbils

Astromòbils lunars

Al llarg de la història hi ha hagut molts tipus d'astromòbils. El primer astromòbil que va aterrar a la Lluna el 1970 va ser el Lunokhod 1[Pas15], que pertanyia a la missió Lluna 17. Enviat pel govern de l'URSS i el programa Lunokhod, aquest robot va ser el primer dirigit per control remot. Tot i que estava pensat per tenir una autonomia de 3 mesos, va arribar a operar durant 1 any. El 1971 va arribar a la Lluna l'astromòbil més recordat per l'occident, el *Lunar Roving Vehicle*(LRV) del programa APOLLO de la NASA[Wil16]. Aquest era un vehicle que transportava astronautes i mostres que aquests recollien per la superfície de la Lluna[sta]. El 1973, l'URSS va llançar el Lunokhod 2[Ter], que va anar amb una missió que va durar 5 mesos, era una versió més pesada i amb més aparells que el Lunokhod 1, una vegada es va cancel·lar el programa, no es varen enviar més rovers fins al 2013. EL 2013, a Xina es va llançar el Yutu per la China National Space Administration(CNSA). Fins a l'actualitat no s'han tornat a enviar Rovers a la Lluna on ara Xina, Índia i la NASA estan preparant prototips.

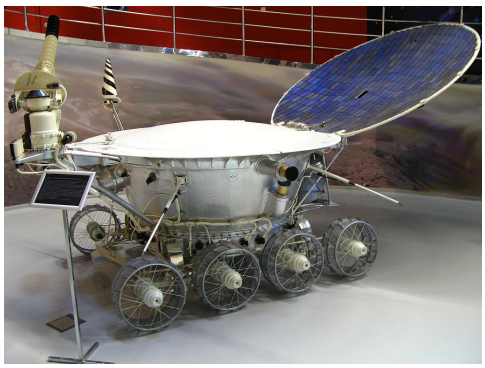

Figura 2.9.: Rèplica del Lunokhod 1

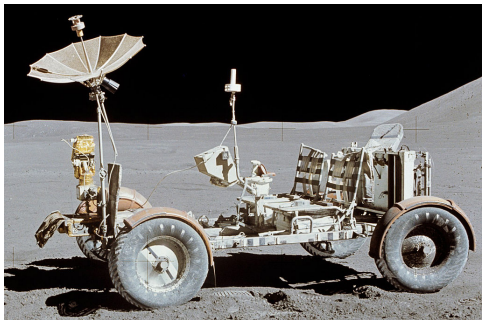

Figura 2.10.: Lunar Rover de la Nasa, permetia transportar astronautes i recollir mostres

Astromòbils enviats a missions a Mart

Una vegada el programa lunar va decaure, el següent va ser explorar el planeta Mart. Des d'aquell moment es van enviar astromòbils, en anglès rover, dels quals es varen fer més famosos el Mars Exploration Rover i el Curiosity, [spa17]. El primers rovers que van arribar a Mart van ser el Soviet Mars 2 i 3, que tenien a dins un petit rover Prop-M el 1971. El 1973, la Unió Soviètica va arribar a Mart el Marksokhod que era una mescla entre robot automàtic i de telecomunicacions.

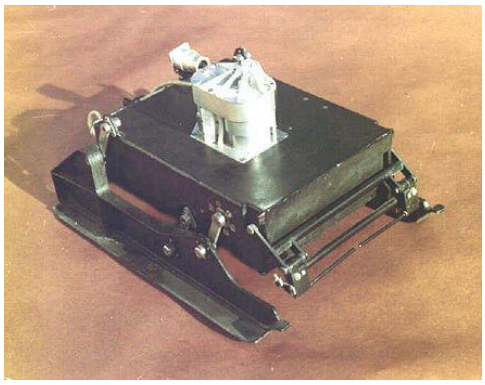

Figura 2.11.: rover Prop-M llançat per el sovier mars 2 i 3

A final del 1996, la NASA va enviar el Mars Pathfinder i va recollir dades sobre els components químics de la superfície de Mart i gairebé 65.000 fotografies. El 2003, l'Agència Espacial Europea va enviar el Beagle 2, que tenia un mecanisme per agafar mostres i desplaçar-se per la superfície de Mart. El 2004, l'Spirit i l'Oportunity varen aterrar a Mart[Sci14b], usant panells solars per agafar energia i recollir dades del planeta, l'Oportunity ostenta el rècord de més quilòmetres recorreguts sobre la superfície del planeta 45, va acabar la seva missió el 2018.

Figura 2.12.: Imatge del Mars pathfinder

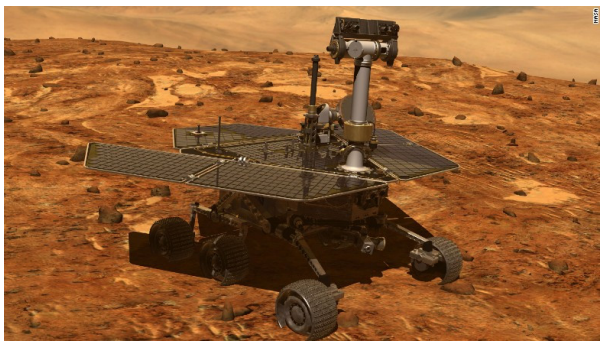

Figura 2.13.: Imatge de l'Opportunity Rover

Finalment, el Curiosity[NAS15] creat per la NASA que circula actualment des del 2010 sobre la superfície de Mart amb l'objectiu de cercar indicis d'algun tipus de mostra que pugui demostrar que en algun moment ha existit vida a Mart a més de buscar aigua i materials[Sci14c].

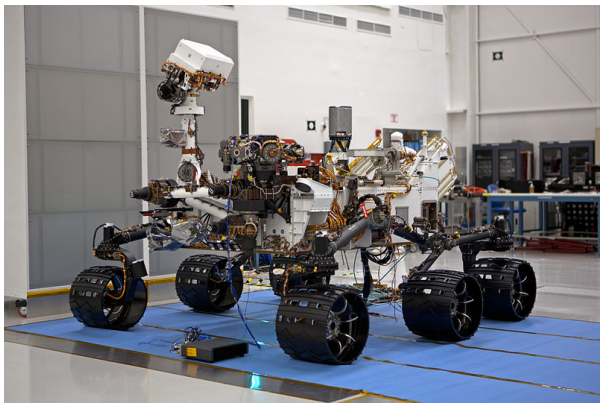

Figura 2.14.: Imatge del Curiosity Rover

2.3.3. Open Source Rover

A fi de fomentar el coneixement sobre aquest tipus de robots, la Nasa va publicar l'Open Source Rover(OSR) amb la intenció de que els instituts americans poguessin tenir una rèplica versió reduïda del Curiosity.

L'OSR és una rèplica del Curiosity a una escala molt reduïda a fi de poder ser construït per instituts[Nas18]. Aquest projecte, publicat per la NASA l' Abril de 2018, és un projecte pensat per arribar als instituts americans amb la finalitat de motivar els joves a aprendre robòtica. Com que està pensat perquè sigui adquirit per un institut, el cost és elevat(2600$), utilitzant materials comercials i de qualitat.

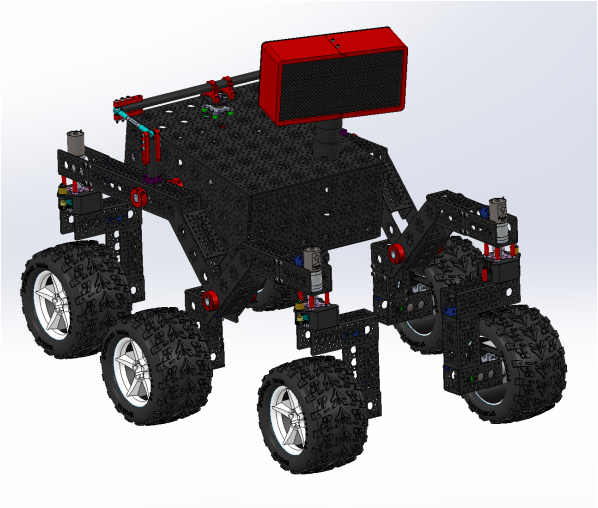

Figura 2.15.: Open Source Rover

L'OSR està format per 3 blocs principals:

1) Estructura: En l'OSR, l'estructura es 100% comprada a Actobòtics i feta de ferro. Reprodueix la suspensió que usa el Curiosity, que porta millor facilitat per estabilitzar a l'hora de superar obstacles.

2) Cervell: S'usa una Raspberry Pi com a cervell del robot, la qual s'encarrega d'executar el software. El software gestiona per una banda el moviment i per una altre les comunicacions que configurant un flag es poden rebre des d'una aplicació o un comandament de Xbox. El sistema es dirigeix mitjançat joysticks físic o virtual des de l'app de mòbil, ambdós usant Bluetooth.

3) Motors: El sistema de motors que usa l'OSR és un sistema basat en els Roboclaw Motor Controller, que permetrà a qualsevol institut augmentar la complexitat del sistema en un futur.

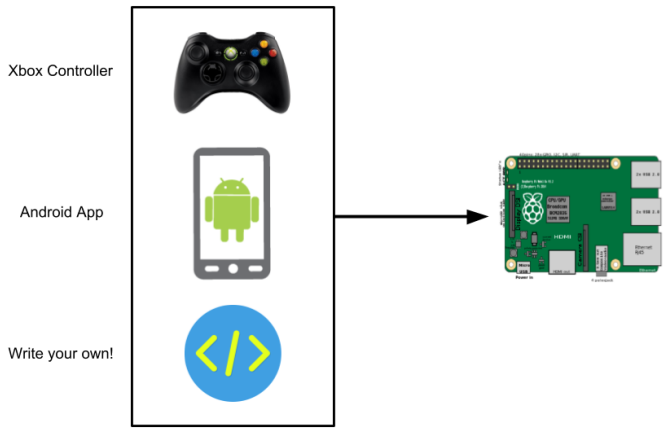

Figura 2.16.: Control de l'OSR

A partir del projecte OSR se n'han creat alternatives o adaptacions més simples amb l'objectiu d'abaratir-ne el cost, arribant a reduir-lo fins a gairebé 2000$. Per exemple, aquest seria el cas del Sawppy Rover[Rog18b], rovers fet a partir de ROS[ROS18], rovers partir de Lego Mindstorm[osr18] o el projecte Open Curiosity amb Arduino [DIY14].

Sawppy Rover

A continuació es farà un anàlisi comparatiu entre l'OSR i el Sawppy Rover [New] [Rog19], en diverses parts del projecte:

1) Estructura: En l' OSR, l'estructura es 100% comprada integrament a Actobòtics i feta de ferro, mentre que al Sawppy Rover, a excepció del xassís d'alumini, la resta de l'estructura està feta amb impressores 3D i per tant, hi ha un abaratiment significatiu respecte l'estructura de l' Open Source Rover. A més, el fet de que sigui construït en de plàstic permet una reducció del pes i, per tant, la possibilitat d'utilitzar motors menys potents.

2) Cervell: S'usen en el 2 casos una Raspberry Pi com a cervell del robot.[Rog18a]. Pel que fa a l'interfície són completament diferents. El Sawppy, usa una interfície web amb joysticks. En canvi l'OSR, pots triar entre joysticks del comandament de XBOX o de l'app de mòbil mitjançant Bluetooth.

3) Motors: La principal diferència entre els 2 projectes, es troba en aquest apartat. En el OSR, s'usen uns actuadors complexos que els *Roboclaws motor controllers* permet que hi hagi una precisió elevada. Ara bé, el Sawppy Rover substitueix aquest sistema per 10 servo motors RC obtenint un resultat equivalent.

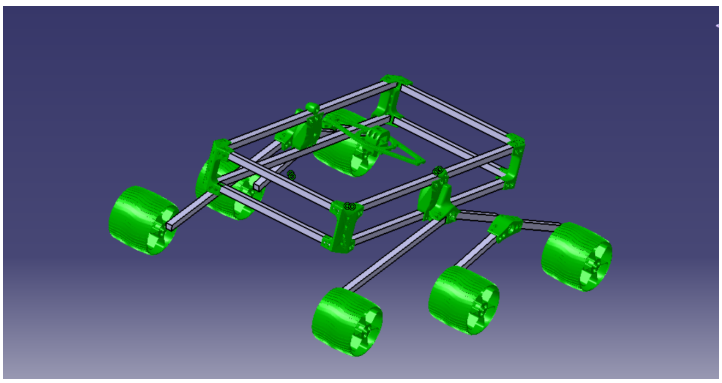

Figura 2.17.: Model aproximat de l'estructura del sawppy rover

3. Objectius

L'objectiu principal d'aquest projecte és la integració d'una eina de reconeixement de la parla per substituir el comandament que portava un robot Open Source. Per una altre banda també interessa aquest reconeixement pugui ser usat en un futur en qualsevol altre sistema.

A banda de l'objectiu principal existeixen diversos objectius secundaris que s'assoliran.

Per començar, ampliar el coneixement sobre els robots en què es basa el projecte i a més a més, dels diferents reconeixements de veu que existeixen i com es duen a terme. Després poder fer un sistema més robust que pugui tenir en compte algun tipus d'errors que es podrien causar en el robot.

Aplicar els coneixements obtinguts durant la titulació d'Enginyeria de Sistemes TIC, i aprofitar la seva transversalitat.

4. Requeriments del sistema

En aquest projecte es substitueix el sistema que existeix en el Sawppy Rover per una interfície de reconeixement de la veu. Per tant, serà necessari fer una equivalència entre les comandes que ja reconeix Julius, per les que existeixen en el Sawppy.

En aquest cas, el sistema del Sawppy, consisteix en una interfície de joysticks, que s'encarreguen de moure el robot en diverses direccions. Pel que fa al nostre sistema volem tenir unes comandes molt clares que permetin al robot fer una sèrie de moviments que puguin ser el més natural possible pels humans. Per tant, s'ha decidit agafar aquestes comandes:

- *up*
- *down*
- *right*
- *left*
- *fast*
- *slow*
- *stop*
- *straight*

Com podem veure en la figura4.1 del moviment del rover, la idea és que sigui un sistema que qualsevol persona pugui simular el que es duria a terme amb el sistema original amb unes comandes molt simples de veu. Poder dir *right* o *left* per girar, *up* o *down* per moure, *fast* o *slow* per limitar la velocitat, *straight* per anar recte i *stop* per parar. Ara bé al ser un sistema de 6 rodes i amb suspensió com l'original, no té sentit que es mogui completament en horitzontal (un rover real no fa aquest moviment), degut a que perdria totes les avantatges que donen les suspensions. Una altra de les coses importants que ha d'assolir el sistema és que sigui robust per tant, voldrem que si el sistema falla, es pari i sigui capaç de gestionar mal contactes amb els motors i el micròfon.

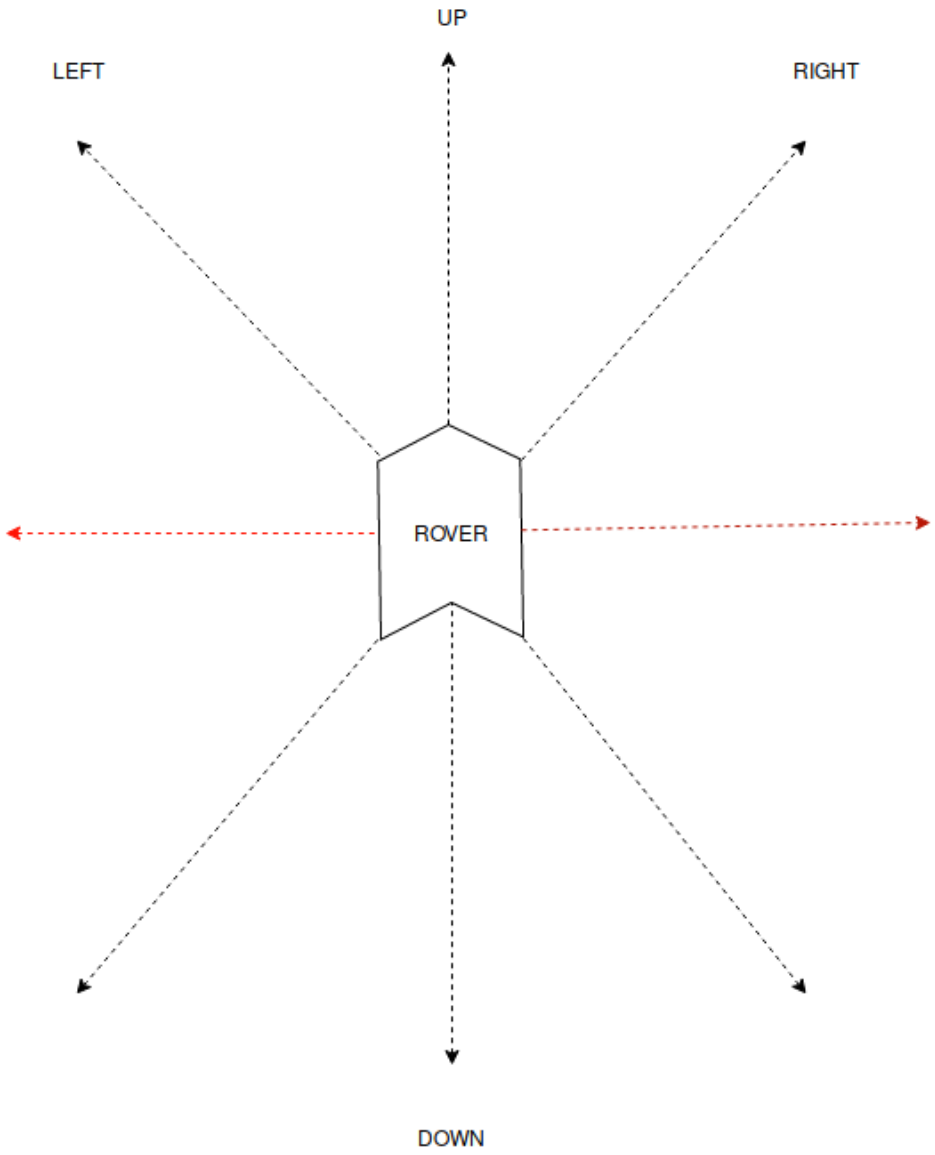

Figura 4.1.: Moviment que podria dur a terme el rover

5. Components

Una vegada repassats els sistemes de reconeixement que existeixen i els diversos rovers Open Source que hi ha, podem començar a prendre decisions importants sobre el sistema que usarem. Primer, el que ens interessa principalment en el reconeixement de la veu, és que sigui un sistema ràpid malgrat que no sigui tan precís com altres. A més a més, ens interessa que ocupi la menor memòria possible per poder posar-ho en entrons de treball amb poca memòria. Per això, el sistema de reconeixement de la parla que usarem serà Julius. Perquè malgrat que no és el més precís sí que és el més ràpid i, perquè només tindrem una interfície per poder comunicar-nos amb el robot. Finalment Julius també té una llibreria en Python que permet integrar amb més facilitat el sistema amb projectes Open Source. Pel que fa al rover, ens interessa que el cost sigui molt menor a l'Open Source Rover i que existeixi la possibilitat de posar-li millores en un futur.

5.1. Sawppy Rover

Malgrat seria molt interessant poder basar-se en l'Open Source Rover publicat per la NASA, el cost és molt elevat per a un individu i usarem el sistema del Sawppy Rover per poder assumir el cost. També, podem fer implementacions sobre aquest sistema per a poder fer-lo més robust. El Sawppy essent un sistema completament Open Source, ens permet modificar aquest al nostre gust. Per tant, poder afegir el reconeixement de la veu modificant el codi ja existent.

5.1.1. Material Sawppy Rover

El Sawppy Rover té moltes opcions pel que fa als motors. Després d'estudiar totes les variables que es presenten i proposar-ne d'altres de diversos tipus, com per exemple la d'usar motors pas a pas o motors DC de molts de tipus, es va decidir que el més correcte seria l'ús d'un sistema de motors que ens pogués aportar la precisió dels pas a pas i la capacitat dels DC. Per tant, usarem els Lewansoul lx-16A, que porten un *encoder* que ens permet saber la posició exacte dels motors i a més l'ús d'un mode DC que ens permet moure els motors a una certa velocitat mantenint un parell motor de 17Kg que ens permet que pugui ser usat per superar obstacles com ho faria qualsevol astromòbil.

Per dur a terme la simulació del Sawppy usarem:

- Raspberry pi 3B
- 10 motors Lewansoul lx-16A
- Driver d'alimentació pels motors.
- Font d'alimentació o bateria.
- Cable usb 2.0 per comunicar la Raspberry Pi amb els motors

5.2. Reconeixement de veu

La interacció amb les maquines i les persones, és el que ha despertat la idea de com es podria arribar a implementar a un robot que accepti comandes de veu i buscar les alternatives per poder du això a terme i quins poden ser els factors diferencials entre aquests. Com ja hem vist a la secció 2.2, ens hem trobat amb diferents alternatives per a poder dur a terme això. Ara bé, ens interessa un reconeixement per a poder ser usat en un robot que simula un astromòbil, per tant, busquem un reconeixement que sigui el més ràpid possible. Si hi hagués algun obstacle que no pot superar o un cràter,es necessari que es pari a l'instant i no es precipiti. Per tant, el sistema de reconeixement que usarem serà Julius, que després d'haver investigat i buscat comparacions amb altres ens trobem que és el més ràpid i que a més a més ocupa menys memòria i és compatible l'ús amb Python.

5.2.1. Detecció de la parla

Una altra part molt interessant és que Julius ens permet crear els fitxers per dur a terme els nostres propis models acústics i de llenguatge. Per tant, crearem els fitxers per dur a terme comandes per a robot. Els fitxers que modificarem seran els .grammar i el .voca. Julius, no té un model de llenguatge per tots els idiomes, només per japonès i des d'aquest Maig el model d'anglès. Ara bé, en ser unes comandes molt simples de direcció no és necessari usar aquests models que l'únic que aconseguirien seria més error i lentitud que no és desitjada.

5.2.2. Model ocults de Markov a Julius

En Julius s'usen els HMM. En aquest cas la llibreria de Julius internament utilitza el HTK per a fer aquesta sèrie de càlculs probabilístics. Nosaltres, podem crear els fitxers .voca i .grammar per compilar i crear models que s'utilitzaran en aquesta llibreria.

5.3. Software

Pel que fa al software, aquest sistema s'aprofitarà part del codi font original del sistema del Sawppy Rover però fent algunes modificacions importants. La primera de totes, serà la desaparició de l'interfície web en el sistema, ja que no ens interessa que es faci mitjançant un router perquè si hi ha una connexió lenta, tardarà molt a arribar el senyal al robot. Per una altra banda, el sistema no és robust per a la connexió amb els motors, ja que es comunica per UART i no volem que per una mala connexió es pugui perdre la comunicació completament amb el robot. Si es produeix un problema amb l'USB i es canvia el *device*, el robot perd la possibilitat de rebre ordres i podria ser crític que es perdi la connexió amb els motors. En l'esquema de la figura 5.1 podem il·lustrar el diagrama de classes amb les funcions i paràmetres més importants que hi ha en el software del Sawppy:

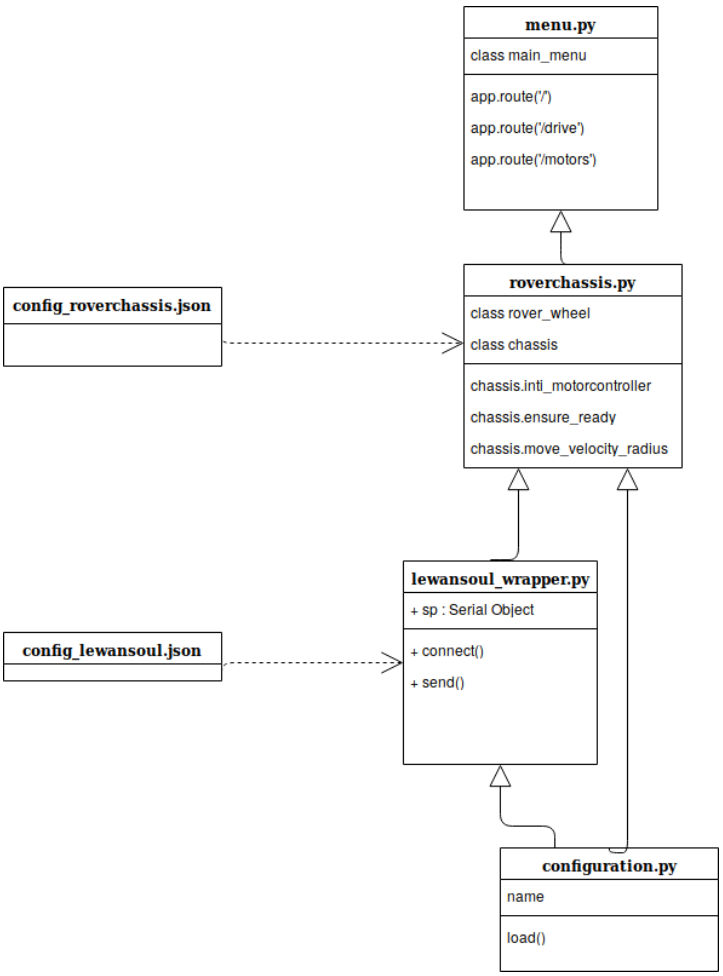

Figura 5.1.: Diagrama de classes Sawppy Rover

6. Implementació del projecte

La idea principal d'aquest projecte és obtenir un sistema de control d'un robot a partir del reconeixement de veu. El seu disseny es divideix en tres parts:

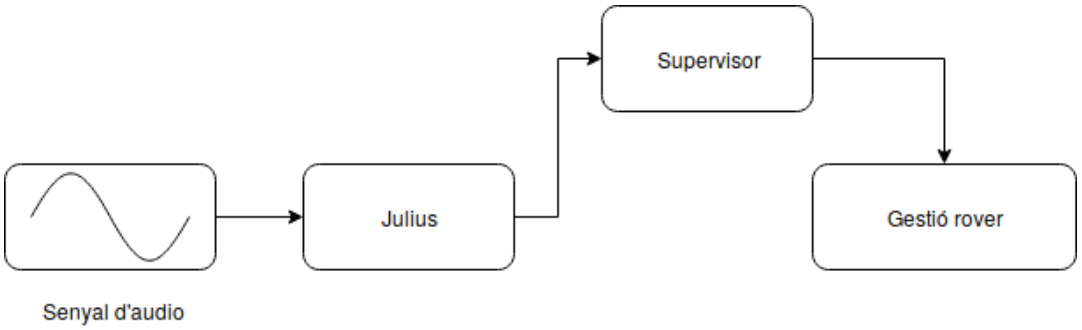

Figura 6.1.: Esquema general del sistema

- Julius: S'encarrega del reconeixement de la parla.
- Supervisor: S'encarrega de gestionar possibles errors del sistema i de enviar missatges a gestió del Rover.
- Gestió rover: S'encarrega de moure el sistema.

Per poder arribar aquí, s'ha centrat el projecte en 3 parts diferents i s'analitzaran una per una per poder explicar com s'ha arribat a la presentació final.

6.1. Software Julius

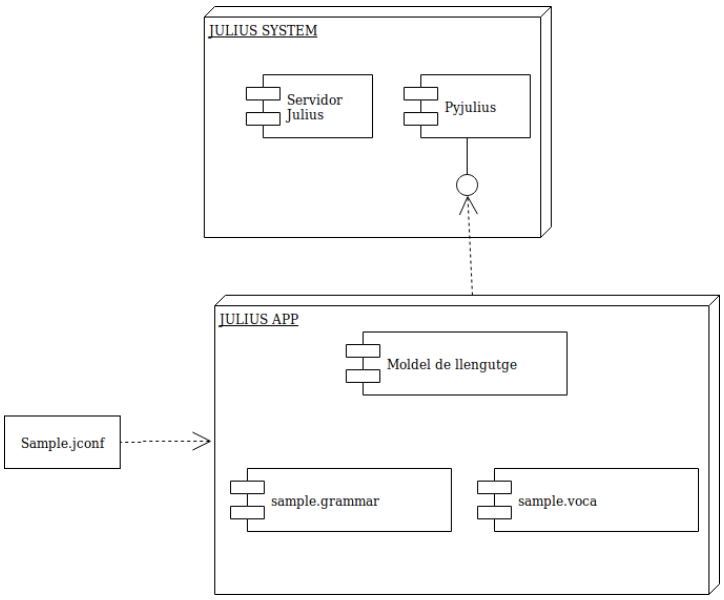

Figura 6.2.: Sistema de reconeixement de veu

Com s'ha descrit anteriorment, s'ha usat Julius per a poder fer el reconeixement de la veu. El primer que fem, és modificar els fitxers font de Julius per a poder fer els models de llenguatge i acústic que volem pel nostre projecte.

```
% NS_B
<s>          sil

% NS_E
</s>          sil

% ACT
RIGHT          r ay t
LEFT           l eh f t
UP           ah p
DOWN           d aw n
FAST           f ae s t
STOP           s t aa p
SLOW           s l ow
STRAIGHT             s t r ey t

% NAME
ROVER          r ow v er
```

Mòdul 6.1: sample.voca

En el fitxer .voca, s'escriu la paraula amb els corresponents fonemes que la conformen. Aquestes paraules i els corresponents fonemes, s'extreuen d'un diccionari creat per VoxForge [org19b]. Addicionalment les paraules són agrupades formant les frases a reconèixer en el fitxer d'extensió .grammar.

```
S : NS_B NAME_T ACTION NS_E
NAME_T: NAME
ACTION: ACT
```

Mòdul 6.2: sample.grammar

En el fitxer .grammar, usarem les etiquetes com per exemple la de nom, prèviament preparades en el .voca. Una vegada formades aquestes frases, és necessari fer una compilació amb el fitxer mkdfa.sl que Julius proporciona per crear els models acústics i de llenguatge que Julius usarà per fer el reconeixement.

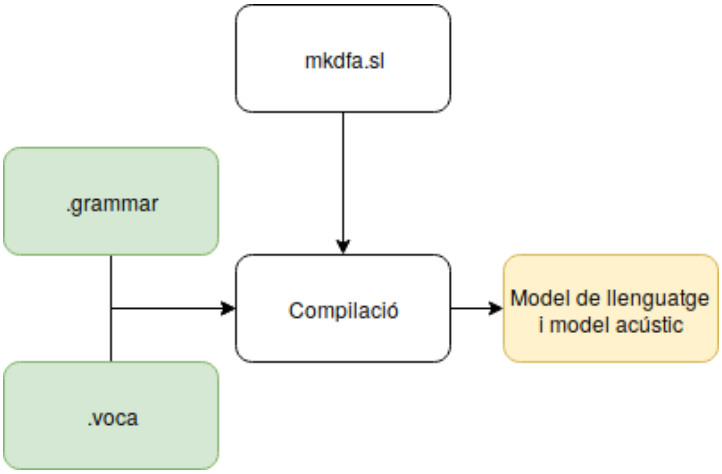

Figura 6.3.: Generació del model acústic i de llenguatge

El sistema de Julius, compta amb moltes opcions per configurar el sistema al teu gust. Les configuracions es guarden en un fitxer d'extensió .jconf. Aquest fitxer internament duu ja preparades configuracions com els nivells de llindar del senyal. Ara bé, no es necessari usar les configuracions del sistema, si executem el sistema en un terminal de Linux per exemple, podem afegir *flags* amb les configuracions que volem i s'aplicaran al sistema.

Finalment, usarem el sistema de Julius com a servidor i mitjançant una connexió, servidor-client enviarem informació i l'extraurà amb Pyjulius en el client.

6.2. Sistema supervisor de Julius

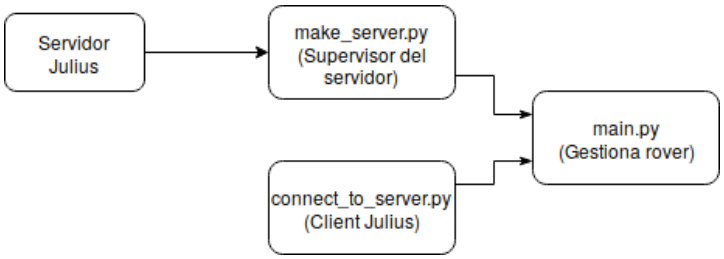

Figura 6.4.: Sistema supervisor de Julius

Pel que fa al micròfon, s'ha hagut de fer una implementació molt més complexa que es complementa amb el sistema per poder dur a terme aquesta feina. Cadascun dels blocs que es veuen a la figura6.4 és equivalent a un procés o subprocés del sistema.

El primer fitxer, make_server.py s'encarrega de crear el servidor en un subprocés i gestionar el que envia per l'stdout. A més a més, obre una pipe amb nom per enviar missatges al gestor. En el cas que rebi un error de l'entrada d'àudio o algun problema que prové del sistema, s'enviarà mitjançant la pipe, al procés gestor(main.py).

```
FIFO = 'mypipe'

print "Creem servidors de Julius"
#Creem el servidor amb un subprocess
try:
    process_servidor = subprocess.Popen("padsp julius -input mic -C /home/pi/
    Desktop/Sample.jconf -module", shell = True, stdout = subprocess.PIPE)
except:
    raise ValueError("No funciona el servidor")

print "Servidor creat"

try:
    os.mkfifo(FIFO)
except OSError as oe:
    if oe.errno != errno.EEXIST:
        print "LA FIFO ja existeix"
        pass

print "Creant pipe si no existeix"
try:
    fifo = open(FIFO, 'w')
except:
    pass
fifo.close()
print "Pipe preparada"

while True:
    output_servidor = str(process_servidor.stdout.readline())
    print output_servidor
    if str(output_servidor[0:21]) == "Error: adin_alsa: una":
        fifo = open(FIFO, 'w')
        fifo.write("Micro out")
```

```
        fifo.close()
        time.sleep(1)#Per donar temps al gestor a executar accions

    if str(output_servidor[0:22]) == "Warning: strip: sample" :
        fifo = open(FIFO,'w')
        fifo.write("Microfon no funciona")
        fifo.close()
        time.sleep(1)#Per donar temps al gestio a executar accions
```

make_server.py

En el fitxer connect_to_server.py es farà una connexió com a client al servidor de Julius, que s'encarregarà de rebre un reconeixement i rebrer-la mitjançant la classe pyjulius amb tota mena d'informació sobre el reconeixement que s'ha dut a terme. Una vegada iniciada la connexió es farà un get(False) a la cua del client de pyjulius, que es quedarà esperant a que el servidor. Finalment, s'obté una nova instància amb una certa informació per poder ser filtrada i enviada mitjançant la mateixa pipe. Per tant, tenim 1 canal amb 2 escriptors.

```
FIFO = 'mypipe'

# Initialize and try to connect
client = pyjulius.Client('localhost', 10500)
try:
    client.connect()
except pyjulius.ConnectionError:
    print 'Start julius as module first!'
    sys.exit(1)

#Es crea la fifo si no existeix
try:
    os.mkfifo(FIFO)
except OSError as oe:
    if oe.errno != errno.EEXIST:
        print "LA FIFO ja existeix"
        pass

# Start listening to the server
client.start()

try:
    while 1:
        try:
            result = client.results.get(False)
            print result
        except Queue.Empty:
            print "Impossible rebre res"
            continue
        if isinstance(result, pyjulius.Sentence):
            fifo = open(FIFO,'w')
            fifo.write(str(result))
            fifo.close()
        else:
            print "S'ha rebut algo incorrecte"

except KeyboardInterrupt:
    print 'Exiting...'
    client.stop()  # send the stop signal
```

```
    client.join()  # wait for the thread to die
    client.disconnect()  # disconnect from julius
```

connect_to_server.py

El fitxer main.py s'encarrega de gestionar els missatges que arriben de les pipes mitjançant una simple lectura i una vegada processat els missatges executarà les accions corresponents.

```
    rover = JuliusRover()
    rover.rover_ready()
    velocity_rover = 20
    angle_rover = 0
    while True:
        with open(pipe) as fifo:
            print "Fifo opened"
            while True:
                try:
                    data = fifo.read()
                    print data
                    if str(data) == "Microfon no funciona":
                        print "Microfon no funciona"
                        rover.stop_motors()

                    if str(data) == "Warning: strip: sample" :
                        print "Microfon no funciona"
                        rover.stop_motors()

                    if str(data) == "rover stop":
                        print "I stop"
                        rover.stop_motors()

                    if str(data) == "rover up":
                        if velocity_rover < 0:
                            velocity_rover = -velocity_rover
```

main.py

6.3. Software Sawppy Rover

- configuration.py: Permet carregar la informació des d'un fitxer .json i retornar el contingut d'aquest
- lewansoul_wrapper.py: Aquest fitxer s'encarrega de fer totes les gestions de la UART per a poder fer la comunicació amb els motors.
- roverchassis.py: En aquest fitxer hi ha 2 classes: la classe roverwheel, que funciona com a controlador per a una sola roda, i la classe roverchasssis, que s'encarrega de gestionar el moviment amb totes les rodes.
- menu.py: Aquest fitxer està format per la classe Julius_Rover que s'encarrega de fer una crida a les capes inferiors i poder dur a terme les comandes assegurant-se que el robot funcionarà.
- make_server.py: Aquest programa s'encarrega d'iniciar el servidor de Julius i gestionar els missatges que envia el servidor.
- connect_to_server.py: Aquest programa s'encarrega de la connexió amb el servidor i de filtrar el que és rep del reconeixement.
- main.py: Aquest programa s'encarrega de rebre missatges de make_server i de connect_to_server.py i executar les accions pertinents en el rover.

6.3.1. Modificacions del software que s'han dut a terme

El software inicial del Sawppy, estava pensat perquè el Rover portés un router i es pogués controlar mitjançant una interfície web. Ara bé, com serà necessària una interfície on sigui el més ràpid possible se substituirà pel reconeixement de la parla. Primer, quan es van haver triat els motors es va posar els motors correctes en el fitxer roverchassis perquè les comandes estiguin adaptades als motors que s'usen en el projecte.

Una vegada testejat i provat el software després d'haver fet aquest canvi, s'ha substituït el mòdul menu.py original per un altre de nou. Aquest mòdul implementa la classe Julius_Rover. Aquesta classe tindrà 3 mètodes, que serviran per dur a terme el moviment i el control dels motors cridant a les capes inferiors.

Finalment, s'han creat 3 fitxers(make_server,connect_to_server i main.py) que ens permeten fer gestió tant del reconeixement de la veu, com del *hardware* que hi ha en el sistema.

Implementació de la robustesa

Per a poder fer robust el programa, s'ha fet un estudi d'on es podrien ocasionar els problemes per possible mals contactes a l'USB o el micròfon.

El que s'ha descobert és que el *driver* del motor, es detecta en el sistema operatiu com a un tipus de dispositiu i amb un id concret.

Per tant, s'ha fet una funció dins el mòdul de lewansoul_wrapper.py que ens permet saber en quin port s'ha connectat segons l'id del dispositiu i poder fer la connexió sèrie i resoldre la problemàtica d'una mala connexió o una desconnexió per haver superat un obstacle i que es pugui tornar a connectar i continuar la seva feina.

```
def connect(self):
  """
  Read serial port connection parameters from JSON configuration file
  and open the port.
  """

  # Read parameter file
  config = configuration.configuration("lewansoul")
  connectparams = config.load()['connect']

  # Open serial port with parameters
  s = serial.Serial()
  s.baudrate = connectparams['baudrate']
  s.port = connectparams['port']#Deixem per tenir un per defecte en el cas que
   no hi hagi cap conectat
  self.change_device(s)
  print "***************"
  print "Port that has changed"
  print s.port
  print "***************"
  s.timeout = connectparams['timeout']
  s.open()

  if s.is_open:
    self.sp = s

def change_device(self, s):
  """
  Function to change device
  """
  ports_instance = serial.tools.list_ports.comports()
  ports_available = list()
  for ports in ports_instance: #vid no canvia fer comprovacio amb id
    if str(ports.vid) == "6790":
      s.port = str(ports.device)
      self.port_change_ok = True
      break
```

lewansoul_wrapper.py

Diagrama final del sistema

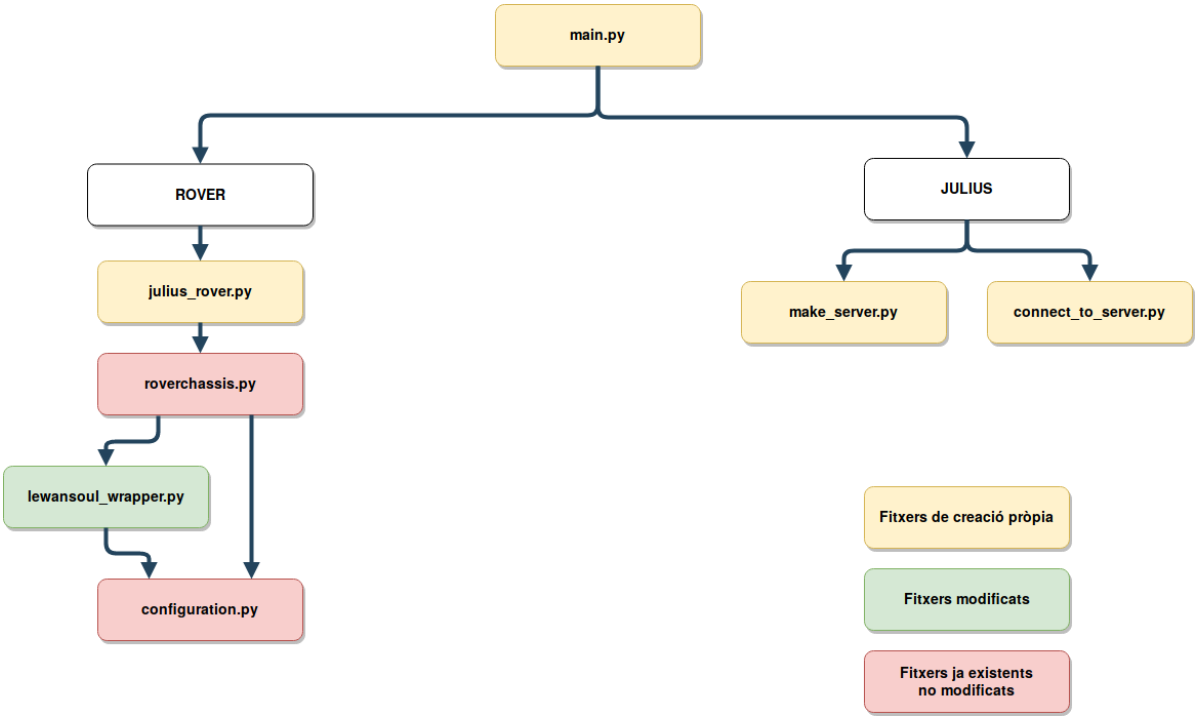

Figura 6.5.: Diagrama de classes del sistema, el codi del qual es pot trobar a l'apèndix

6.4. Conclusió

La tecnologia és un del àmbits que més s'ha desenvolupat durant les últimes dècades, com s'ha pogut veure durant el repàs de la història tant del reconeixement de la parla com dels astromòbils. El reconeixement de la parla és una de les noves tendències que sembla que en un futur serà la manera més còmode i natural per interactuar amb les màquines. Aquest sistema és un camp on malgrat que s'ha fet una gran feina, pot arribar a evolucionar molt més. També l'arribada i l'evolució de noves tecnologies emergents com són la intel·ligència artificial i el *machine learning* ajudaran a fer que els sistemes cada cop siguin més precisos.

Per una altra banda, s'ha repassat la història dels viatges espacials i dels vehicles d'exploració espacial, que fins i tot ara es segueixen desenvolupant. Per això, associacions com la NASA han creat projectes com l'Open Source Rover amb el fi de motivar a generacions futures.

Integrar el reconeixement de veu al sistema ha requerit un anàlisi en profunditat del software i les eines existents. Aquest estudi ha permès dissenyar un sistema robust basat en programari lliure.

S'ha hagut de substituir tota la interfície existent i veure com funcionava per a poder fer una equivalència amb comandes de veu. A més a més de l'estudi d'aquesta eina de reconeixement de la parla que quant es coneix bé pot arribar a ser molt útil per a usar en qualsevol sistema.

Mitjançant el coneixement que s'ha adquirit durant els anys a la carrera s'ha pogut dur a terme tot amb facilitat usant diversos processos que s'encarreguen de dur a terme la gestió.

Una vegada realitzades les proves amb el sistema complet i pensant maneres de millorar, es van fer modificacions que van permetre que el sistema pogués ser el més robust possible. En aquest cas, centrat en possibles desconnexions als ports usb, per això s'ha creat un sistema que es capaç de conèixer els possibles problemes i avisar o dur a terme accions per resoldre els diversos problemes que poden produir-se, fent que sigui un sistema robust.

Aquest treball m'ha servit per a poder dur a terme un sistema complet que engloba molts coneixements del grau, ja que engloba des d'estudiar el *hardware* del robot, el software que em creat i les comunicacions i sistemes interns que usem per a gestionar aquesta robustesa.

A més, el fet d'usar LaTeXper redactar el treball, m'ha suposat un gran aprenentatge d'una eina que no coneixia massa.

Algunes de les línies futures per aquest treball serien:

- Acabar de fer la carcassa del Sawppy per poder fer conducció del rover.
- Implementar un software per poder reiniciar automàticament el rover en el moment que es torni a connectar el micròfon.
- Afegir elements al Sawppy que es puguin connectar i controlar al sistema, com per exemple, una càmera i un braç robòtic.
- Afegir al rover més sensors perquè el robot pugui dur a terme accions autònomes sense rebre comandes.

Bibliografia

[Blu14] Phil Blunsom. *Hidden Markov Models*. Vers. definitiva. 2014. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.1016&rep=rep1&type=pdf.

[Bof94] Pau Bofill. *Xarxes Neurals Artificials*. Vers. definitiva. 1994. URL: https://publicacions.iec.cat/repository/pdf/00000120/00000074.pdf.

[BR04] B.H.Juang i Lawrence R. Rabiner. *Història del Reconeixadors de veu automàtics*. Vers. definitiva. 2004. URL: https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf.

[Bri14] Britannica. *Primers viatges espacials*. 2014. URL: https://www.britannica.com/technology/spacecraft (cons. 21-02-2019).

[CMU18] CMUSphinx. *Llibreria CMUSphinx*. Ang. 2018. URL: https://cmusphinx.github.io/ (cons. 21-02-2019).

[DIY14] DIY. *Open Curiosity*. 2014. URL: http://diymakers.es/opencuriosity-robot-explorador-de-marte-con-arduino/ (cons. 21-02-2019).

[elS16] elSoftwareLibre. *Reconeixement en Ubuntu amb Google2Ubuntu*. Ang. 2016. URL: https://elsoftwarelibre.com/2016/03/01/reconocimiento-por-voz-en-linux-con-google2ubuntu/ (cons. 21-02-2019).

[Gel18] Janna Escur i Gelabert. *Exploring Automatic Speech Recognition with TensorFlow*. Vers. definitiva. 2018. URL: https://upcommons.upc.edu/handle/2117/117342.

[git19] github. *Repositori de Github de Julius*. Ang. 2019. URL: https://github.com/julius-speech/julius (cons. 21-02-2019).

[Goo18] Google. *Llibreria TensorFlow*. Ang. 2018. URL: https://github.com/tensorflow/models/tree/master/research/deep_speech (cons. 21-02-2019).

[ICT] ICT. *Procesamiento Automático de Voz*. Cast. URL: http://ict.udlap.mx/people/ingrid/Clases/IS579/ProcVoz.html (cons. 28-03-2019).

[ins] instructables. *Controlar GPIO Raspberry Pi amb Amazon Echo i python*. Ang. URL: https://www.instructables.com/id/Control-Raspberry-Pi-GPIO-With-Amazon-Echo-and-Pyt/ (cons. 21-02-2019).

[ins18] instructables. *Google home per controlar una Raspberry Pi*. Ang. 2018. URL: https://www.raspberrypi.org/forums/viewtopic.php?t=175261 (cons. 21-02-2019).

[JJFG09] Stanley I. Grossman S. i José Job Flores Godoy. *Aplicaciones Álgebra lineal*. Cast. Mc Graw Hill, 2009. 746 **pagetotals**. ISBN: 9786071507600.

[JM18] Daniel Jurafsky i James H. Martin. *N-gram Language Models*. 2018. URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf (cons. 15-04-2019).

[LME10] Mumtaj Begam Lindasalwa Muda i I. Elamvazuthi. *Voice Recognition Algorithm using Mel Frequency Cepstral Corfficient(MFCC) and Dynami Time Wrapping(DTW) Techniques*. Vers. definitiva. 2010. URL: https://arxiv.org/pdf/1003.4083.pdf.

[Mar09] Alberto Martos. *Breve historia de la carrera espacial. La carrera espacial*. Cast. Spain: Nowtilus, 2009. 337 **pagetotals**. ISBN: 9788497637657.

[Moz19a] Mozilla. *Archius font de Deep Speech*. Ang. 2019. URL: https://github.com/mozilla/voice-web (cons. 21-02-2019).

[Moz19b] Mozilla. *Common Voice,base de dades de Deep Speech*. Ang. 2019. URL: https://voice.mozilla.org/ca (cons. 21-02-2019).

[NAS15] NASA. *Hitòria del Curiosity*. Cast. 2015. URL: https://www.nasa.gov/mission_pages/msl/overview/index.html (cons. 28-03-2019).

[Nas18] California Institute of Technology Nasa. *Open Source Rover*. 2018. URL: https://opensourcerover.jpl.nasa.gov/#!/home (cons. 18-02-2019).

[New] *Viabilitat Sawppy Rover*. 2018. URL: https://newscrewdriver.com/2018/05/09/is-sawppy-the-rover-feasible/ (cons. 18-02-2019).

[Nil13] Tobias Nilson. *Software de reconeixement de la veu*. Vers. definitiva. 2013. URL: http://www.diva-portal.org/smash/get/diva2:623908/FULLTEXT01.pdf.

[org19a] Julius org. *Pàgina oficial de Julius*. Ang. 2019. URL: http://julius.osdn.jp/en_index.php?q=index-en.html#whats_new (cons. 21-02-2019).

[org19b] VoxForge org. *Pàgina oficial VoxForge*. Ang. 2019. URL: http://www.voxforge.org/es (cons. 27-05-2019).

[osr18] osr.org. *Variacions Open Source Rover*. 2018. URL: https://osr.org/blog/kids/nasa-build-your-own-rover-or-lego-rover/ (cons. 21-02-2019).

[Pas15] Pascual. *Història dels Rovers*. Cast. 2015. URL: http://www.esascosas.com/Aficiones-Temas/aficiones/astronautica/vehiculos-planetarios/ (cons. 28-03-2019).

[pi17] Raspberry pi. *Alexa skill per controlar la Raspberry Pi*. Ang. 2017. URL: https://www.raspberrypi.org/forums/viewtopic.php?t=175261 (cons. 21-02-2019).

[Ram] Jose Manuel Ramírez. *Com usar Homekit d'Apple*. Ang. URL: https://www.jmramirez.pro/tutorial/homebridge-con-homekit/ (cons. 21-02-2019).

[Res] Speech Recognition Reseasrchers.(n.d.) *Kaldi Speech recognition*. Ang. URL: https://cmusphinx.github.io/ (cons. 21-02-2019).

[Rit] S. Ritchie. *Mycroft*. Ang. URL: https://cmusphinx.github.io/ (cons. 21-02-2019).

[Riv10] Néstor Pérez Rivero. *Aplicacions de Reconeixement de la Parla. Sistemes de Respeaking III*. Vers. definitiva. 2010. URL: https://ddd.uab.cat/pub/trerecpro/2010/hdl_2072_115155/PFC_NestorPerezRivero.pdf.

[Rog18a] Roger. *Discució Sawppy Rover*. 2018. URL: https://www.meetup.com/es-ES/SGVTech/events/zvpphlyzdbsb/ (cons. 18-02-2019).

[Rog18b] Roger. *Explicació Sawppy Rover*. 2018. URL: https://newscrewdriver.com/category/projects/sawppy-the-rover/ (cons. 18-02-2019).

[Rog19] Roger. *Fitxers Sawppy Rover*. 2019. URL: https://github.com/Roger-random/SGVHAK_Rover (cons. 18-02-2019).

[ROS18] ROS.org. *Sistema ROS*. 2018. URL: http://www.ros.org/ (cons. 21-02-2019).

[Sci14a] Science. *Carrera espacial*. 2014. URL: http://www.juventudtecnica.cu/contenido/sputnik-1-inicio-carrera-espacial-0 (cons. 21-02-2019).

[Sci14b] Science. *Mars History NASA*. 2014. URL: https://www.nasa.gov/feature/50-years-ago-mariner-6-and-7-off-to-mars (cons. 21-02-2019).

[Sci14c] Science. *Mars Rover*. 2014. URL: https://science.sciencemag.org/content/343/6169/386 (cons. 21-02-2019).

[Sci14d] Science. *Primers viatges espacials*. 2014. URL: https://elbigoteobsceno.com/los-primeros-viajes-espaciales/ (cons. 21-02-2019).

[Shm18] N. Shmyrev. *Repositori CMUSphinx*. Ang. 2018. URL: https://cmusphinx.github.io/ (cons. 21-02-2019).

[spa17] space.com. *Hitòria del Rover de Mart*. Cast. 2017. URL: https://www.space.com/13558-historic-mars-missions.html (cons. 28-03-2019).

[sta] starchild. *Vehicle Lunar*. Cast. URL: https://starchild.gsfc.nasa.gov/docs/StarChild_Spanish/docs/StarChild/space_level2/apollo15_rover.html (cons. 28-03-2019).

[Ter] *Historia de los astromóviles en la Luna*. Cast. 2017. URL: http://www.hablandodesdelaatalaya.com/2017/10/historia-de-los-astromoviles-en-la-luna.html (cons. 28-03-2019).

[Wil16] Dr. David R. Williams. *Hitòria dels rovers lunars*. Cast. 2016. URL: https://www.nasa.gov/mission_pages/msl/overview/index.html (cons. 28-03-2019).

Part II.

Apèndixs

Apèndixs del document

```
S : NS_B NAME_T ACTION NS_E
NAME_T: NAME
ACTION: ACT
```

Mòdul 1: sample.grammar

```
% NS_B
<s>         sil

% NS_E
</s>         sil

% ACT
RIGHT         r ay t
LEFT          l eh f t
UP          ah p
DOWN          d aw n
FAST          f ae s t
STOP          s t aa p
SLOW          s l ow
STRAIGHT            s t r ey t

% NAME
ROVER         r ow v er
```

Mòdul 2: sample.voca

```
#
# Sample Jconf configuration file
# for Julius library rev.4.3
#
# 1) Options can also be specified in command line option.
#    The values are default values in Julius.
# 2) For file name, relative path must be relative to THIS FILE.
# 3) Texts after '#' at each line will be ignored.  If you want to specify
#    '#', use '\#'.
# 4) Each line should be no longer than 512 bytes.
#
#

#############################################################################
#### JULIUS APPLICATION OPTION (not a part of JuliusLib)
#############################################################################
#-outfile     # save each result in separate file
#-separatescore     # print AM / LM scores separately
#-callbackdebug     # print callback names at call for debug
#-charconv from to    # print with character set conversion
#-nocharconv      # disable "-charconv"
```

```
#-module      # start in module mode
#-record dir      # record each inputs into dir
#-logfile file      # redirect logs to file
#-nolog       # disable all logs
#-help        # print help, and exit


#################################################################
#### GLOBAL OPTIONS
#################################################################
####
#### Misc. Options
####
#-C jconffile     # include jconf file at here
#-version     # print version info to stderr, and exit
#-setting     # print version info to stderr, and exit
#-quiet       # output less log
#-debug       # output more log for debug
#-check wchmm     # for debug, enter debug shell mode
#-check trellis     # for debug, enter debug shell mode
#-check triphone     # for debug, enter debug shell mode
#-demo        # same as "-quiet -progout"

####
#### Stream Input
####

## Feature vector input
#-input mfcfile     # feature vector in HTK parameter file
#-input htkparam     # (same as "mfcfile")
#-input outprob     # outprob vector in HTK parameter file
#-input vecet     # feature / outprob vector via network client

## Raw audio input
#-input mic         # live microphone
#-input rawfile     # wavefile
#-input file      # (same as "rawfile")
#-input stdin     # waveform from standard input
#-input adinnet     # waveform via network client
#-input netaudio                # DatLink server
#-input oss     # OSS API input (if available)
#-input alsa      # ALSA API input (if available)
#-input esd     # ESounD daemon input (if available)
#-input portaudio   # PortAudio API
#-input pulseaudio    # PulseAudio API


#-filelist filename   # input file list
#-notypecheck     # does not check parameter type of input
#-48        # 48kHz sampling > 16kHz conv. (16kHz only)
#-NA devname      # hostname for DatLink server
#-adport 5530     # port number for adinnet
#-nostrip     # do not strip zero samples
#-zmean       # remove DC offset (use long input average)
#-nozmean     # disable "-zmean" specified before

####
```

```
#### Audio Input Scaling
####
#-lvscale 1.0     # input level scaling factor (1.0 = disable)

####
#### Speech segment detection by level and zero-cross
####
#### default: on for microphone, off for other sources
####
#-cutsilence      # detection on
#-nocutsilence      # detection off
#-lv 2000     # level threshold (0-32767)
#-zc 60       # zero-cross threshold (times in sec.)
#-headmargin 300    # head silence margin (msec)
#-tailmargin 400    # tail silence margin (msec)
#-chunk_size 1000   # processing segment unit length in samples
#-rejectshort 0     # reject shorter input (msec)
#-rejectlong -1     # reject longer input (msec) -1 to disable

####
#### Input rejection by average power (EXPERIMENTAL)
####
####  This will be enabled by "--enable-power-reject" on compilation.
####  Should be used with Decoder VAD or GMM VAD.
####  Valid for real-time input only.
####
#-powerthres 9.0    # reject input by avg. energy

####
#### Gaussian Mixture Model
####
#### GMM will be used for input rejection by accumurated score, or
####  for GMM-based frontend VAD when "--enable-gmm-vad" specified.
####
####  NOTE: If you use MFCC for the GMM which is different from AM, you
####  should also set the parameters like other AM with an option "-AM_GMM".
####  If "-AM_GMM" is not used, Julius assume GMM to use the same
####  parameter as the first AM.
####
#-gmm hmmdefs      # GMM definitions in HTK format
#-gmmnum 10     # num of Gaussians to be computed per GMM
#-gmmreject string    # comma-separated list of GMM name to reject
#### GMM_VAD
#-gmmmargin 20      # head margin for GMM based VAD in frames

####
#### Decoding option
####
#### Real-time processing means concurrent processing of MFCC computation
#### and 1st pass decoding.  By default, real-time processing on the
#### 1st pass is on for microphone / adinnet / netaudio input, and
#### off for others.
####
#-realtime      # force real-time processing
#-norealtime      # force non real-time processing

####
```

```
#### Plug-in
####
#### See plugin/00readme.txt for detail
####
#-plugindir ./plugin:/usr/local/share/julius/plugins


#################################################################
#### INSTANCE DEFINITION FOR MULTI DECODING
#################################################################
####
#### The following three argument will create a new configuration set
#### with default parameters, and switch current set to it.  Jconf
#### parameters specified after the option will be set into the current
#### set.
####
#### To do multi-model decoding, these argument should be specified at
#### the first of each model / search instances with different names.
#### Any options before the first instance definition will be IGNORED.
####
#### When no instance definition is found (as older version of Julius),
#### all the options are assigned to a default instance named "_default".
####
#### Please note that decoding with a single LM and multiple AMs is not
#### fully supported.  For example, you may want to construct the
#### jconf file as this:
####
####  -AM am_1 -AM am_2
####  -LM lm (LM spec..)
####  -SR search1 am_1 lm
####  -SR search2 am_2 lm
####
#### This type of model sharing is not supported yet, since some part
#### of LM processing depends on the assigned AM.  Instead, you can
#### get the same result by defining the same LMs for each AM, like this:
####
####  -AM am_1 -AM am_2
####  -LM lm_1 (LM spec..)
####  -LM lm_2 (same LM spec..)
####  -SR search1 am_1 lm_1
####  -SR search2 am_2 lm_2
####

## Create a new AM configuration set, and switch current to it.
## You should give a unique name.
#-AM name

## Create a new LM configuration set, and switch current to it.
## You should give a unique name.
#-LM name

## Create a new Search configuration set with AM and LM, and switch
## current to it.  AM and LM name can be either name or ID number.
#-SR name am_name_or_id lm_name_or_id

## Switch current AM to special one reserved for GMM, to specify
## analysis parameter for GMM.  Be sure not to confuse with normal AM
```

```
## configuration.
# -AMGMM

## When using instance declarations, global options should be placed
## at top before any instance declaration, or after this option below.
## This option is only a switcher and can be used anywhere anytime.
# -GLOBAL

## This option disables the strict section checkings and back to 4.0
# -nosectioncheck

#############################################################################
#### LANGUAGE MODEL (-LM)
#############################################################################
####
#### Only one type of LM can be specified for a LM configuration.
####

####
#### N-gram
####
#-d binary_ngram_file   # N-gram in Julius binary format
#-nlr ngram      # forward (left-to-right) N-gram
#-nrl rev_ngram      # backward (right-to-left) N-gram
#-v dictfile      # word dictionary
## param.
#-silhead "<s>"      # beginning-of-sentence (silence) word
#-siltail "</s>"     # end-of-sentence (silence) word
#-mapunk "<unk>"     # word to which unknown words should be mapped
#-iwspword      # add a pause word to the dictionary
#-iwspentry "<UNK> [sp] sp sp"  # word that will be added by "-iwspword"
#-sepnum 150      # num of high freq words to linearize
#-adddict dictfile              # append additional word dictionary
#-addword entry                 # append additional word entry

####
#### Grammar
####
#### "-gram", "-gramlist" can be used multiple times
#-gram gramprefix    # (comma-separated list of) grammar file prefix
#-gramlist txtfile    # text file containing grammar prefixes
#-dfa dfafile -v dictfile # specify DFA and dictionary separately
#-nogram      # reset all grammar list already specified

####
#### Isolated Word
####
#-w dictfile      # word dictionary
#-wlist txtfile     # text file containing dictionaries
#-nogram      # reset all dictfiles already specified
## param.
#-wsil silB silE NULL   # head / tail silence models to be appended

####
#### User-defined LM
####
#-userlm      # declare to use user LM defined in program
```

```
####
#### misc LM option
####
#-forcedict      # skip error word entries (no stop on error)

###########################################################################
#### ACOUSTIC MODEL (-AM)  (-AM_GMM)
###########################################################################
####
#### Acoustic analysis parameters are included in this section, since
#### the AM defines the required parameter.  You can use different MFCC
#### type for each AM.
#### For GMM, the same parameter should be specified after "-AM_GMM"
####
#### When using multiple AM,  the values of "-smpPeriod", "-smpFreq",
#### "-fsize" and "-fshift" should be the same among all AM.
####

## Acoustic model
#-h hmmfile      # acoustic HMM (ascii or Julius binary)
#-hlist logicaltri     # HMMList to map logical phone to physical
#-tmix 2       # # of mixture to compute in a mixture PDF
#-spmodel "sp"       # name of a short-pause silence model
#-multipath      # force enable MULTI-PATH model handling
#-gprune {safe|heuristic|beam|none|default} # Gaussian pruning method
#-iwcd1 {max|avg|best 3}  # Inter-word triphone approximation method
#-iwsppenalty -1.0     # pause insertion penalty for "-iwsp"
#-gshmm hmmfile      # HMM for Gaussian mixture selection
#-gsnum 24       # Threshold number of HMM for gshmm

## Analysis
#-smpPeriod 625      # sampling period (ns) (= 10000000 / smpFreq)
#-smpFreq 16000      # sampling rate (Hz)
#-fsize 400      # window size (samples)
#-fshift 160       # frame shift (samples)
#-preemph 0.97       # pre-emphasis coef.
#-fbank 24       # number of filterbank channels
#-ceplif 22      # cepstral liftering coef.
#-rawe         # use raw energy
#-norawe       # disable "-rawe" (this is default)
#-enormal      # normalize log energy
#-noenormal      # disable "-enormal" (this is default)
#-escale 1.0       # scaling log energy for enormal
#-silfloor 50.0      # energy silence floor in dB for enormal
#-delwin 2       # delta window (frames)
#-accwin 2       # acceleration window (frames)
#-hifreq -1      # cut-off hi frequency (Hz) (-1: disable)
#-lofreq -1      # cut-off low frequency (Hz) (-1: disable)
#-zmeanframe       # frame-wise DC offset removal (same as HTK)
#-nozmeanframe       # disable "-zmeanframe" (this is default)

## Cepstral mean / variance normalization
#-cmnload filename     # load initial cep. mean / variance on startup
#-cmnsave filename     # save cep. mean / variance at each input end
#-cmnupdate      # update beginning cep. data at each input
#-cmnnoupdate      # keep initial mean, disable "-cmnupdate"
```

```
#-cmnmapweight 100.0    # weight for MAP-CMN
#-cvn       # enable variance normalization

## Vocal tract length normalization (VTLN)
#-vtln 1.0 300 4800   # enable VTLN (alpha, lowerfreq, upperfreq)

## Spectral subtraction (default: disabled)
#-sscalc      # do SS, estimate noise from head sil
#-sscalclen 300     # length of head silence for "-sscalc" (msec)
#-ssload filename   # do SS, load noise spectrum saved by "mkss"
#-ssalpha 2.0     # alpha coef. for spectral subtraction
#-ssfloor 0.5     # spectral floor coef.

## Others
#-htkconf configfile    # load analysis settings from HTK Config file

###############################################################################
#### RECOGNIZER (-SR)
###############################################################################
####
#### Default values for beam width and LM weights will change
#### according to compile-time setup of JuliusLib and model specification.
#### Please see the startup log for the actual values.
####

####
#### parameter (common)
####
#-inactive      # start this process with inactive status
#-1pass       # perform only the 1st pass, omit 2nd pass
#-no_ccd      # switch off the phone context dependency
#-force_ccd     # force on the phone context dependency
#-cmalpha 0.05      # CM alpha value
#-iwsp        # append a skippable sp at all word ends
#-transp 0.0      # transition penalty for transparent words

####
#### parameter (1st pass)
####
#-lmp weight penalty    # LM weight and word insertion penalty (pass1)
#-penalty1 penalty    # word insertion penalty for grammar (pass1)
#-b width     # beam width (# of nodes)
#-bs score                       # beam width (score)
#-nlimit 3      # with enable-wpair-nlimit, set max N at nodes
#-progout     # progressive output while decoding
#-proginterval 300    # output interval in msec for "-progout"

####
#### parameter (2nd pass)
####
#-lmp2 weight penalty   # LM weight and word insertion penalty (pass2)
#-penalty2 penalty    # word insertion penalty for grammar (pass2)
#-b2 width      # envelope beam width of 2nd pass (#word)
#-sb 80.0     # envelope score width at 2nd pass
#-s 500       # hypotheses stack size on 2nd pass (#hypo)
#-m 2000      # hypotheses overflow threshold (#hypo)
#-n n       # num of sentences to find
```

```
#-output 1      # num of sentences to output as result
#-lookuprange 5     # hypo. lookup range at word expansion (#frame)
#-looktrellis     # expand only trellis words in grammar
#-fallback1pass     # output 1st pass result when 2nd pass fails

####
#### short-pause segmentation (and decoder-based VAD)
####
#-spsegment     # enable sp segmentation (or decoder VAD)
#-spdur 10      # # of frames to detect a short pause
#-pausemodels string    # comma-separated pause model names
#### for decoder-VAD
#-spmargin 40     # backstep margin at trigger up (frame)
#-spdelay 4     # decision delay at trigger up (frame)

####
#### lattice output
####
#-lattice     # output result in word graph (aka -graphout)
#-graphrange 0      # merge same words nearby, -1 to disable merge
#-graphcut 80     # graph depth cut threshold (in depth)
#-graphboundloop 20   # max itertations for boundary adjustment loop
#-graphsearchdelay    # activate an alternate generation algorithm
#-nographsearchdelay    # disable "-graphsearchdelay"

####
#### confusion network output
####
#-confnet     # enable confusion network output
#-noconfnet     # disable confusion network output

####
#### multi-grammar output (for grammar and isolated word)
####
#-multigramout      # output max hypo for each grammar
#-nomultigramout    # disable "-multigramout"

####
#### forced alignment
####
#-walign      # enable alignment for result at word level
#-palign      # enable alignment for result at phoneme level
#-salign      # enable alignment for result at state level

####
#### misc.
####
# !!!!!! VoxForge change
#-outprobout filename   # save computed outprob vectors to HTK file (for debug)
# !!!!!!

# VoxForge configurations:
-dfa grammar/sample.dfa
-v grammar/sample.dict
-h acoustic_model_files/hmmdefs
-hlist acoustic_model_files/tiedlist
-spmodel "sp"    # HMM model name
```

```
-multipath
-gprune safe
-iwcd1 max
-iwsppenalty -70.0  # transition penalty for the appended sp models
-smpFreq 16000    # sampling rate (Hz)
-iwsp     # append a skippable sp model at all word ends
-penalty1 5.0
-penalty2 20.0
-b2 200                  # beam width on 2nd pass (#words)
-sb 200.0   # score beam envelope threshold
-n 1
# !!!!!!
```

Mòdul 3: Sample.jconf

```
"""
MIT License

Copyright (c) 2018 Roger Cheng

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""
import json

class configuration:
  def __init__(self, name):
    self.name = name

  def load(self):
    """
    Loads the configuration file. Filename format is based on the name given
    in the constructor. Prepended by "config_" with ".json" suffix. If all
    goes well, returns a dictionary of configuration parameters.
    """
    filename = "config_" + self.name + ".json"
    filehandle = open(filename, 'r')
    filecontent = filehandle.read(32*1024) # Config files should be << 32kB
    filehandle.close()

    # TODO: Validate JSON data against schema
    return json.loads(filecontent)
```

Mòdul 4: configuration.py

```
{
  "connect": {
    "baudrate": 115200,
    "port": "/dev/ttyUSB0",
    "timeout": 0.5
  }
}
```

Mòdul 5: config_lewansoul.json

```
[
  {
    "name": "front_left",
    "rolling": [
      "lewansoul",
      25,
      500,
      false
    ],
    "steering": [
      "lewansoul",
      23,
      465,
      false
    ],
    "x": -9.125,
    "y": 11.375
  },
  {
    "name": "front_right",
    "rolling": [
      "lewansoul",
      27,
      500,
      true
    ],
    "steering": [
      "lewansoul",
      29,
      515,
      false
    ],
    "x": 9.125,
    "y": 11.375
  },
  {
    "name": "mid_left",
    "rolling": [
      "lewansoul",
      21,
      500,
      false
    ],
    "steering": null,
```

```
    "x": -10.375,
    "y": 0
  },
  {
    "name": "mid_right",
    "rolling": [
      "lewansoul",
      22,
      500,
      true
    ],
    "steering": null,
    "x": 10.375,
    "y": 0
  },
  {
    "name": "rear_left",
    "rolling": [
      "lewansoul",
      20,
      500,
      false
    ],
    "steering": [
      "lewansoul",
      24,
      500,
      false
    ],
    "x": -9,
    "y": -10
  },
  {
    "name": "rear_right",
    "rolling": [
      "lewansoul",
      28,
      500,
      true
    ],
    "steering": [
      "lewansoul",
      26,
      530,
      false
    ],
    "x": 9,
    "y": -10
  }
]
```

Mòdul 6: config_roverchassis.json

```
"""
MIT License

Copyright (c) 2018 Roger Cheng
```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""
import math
import logging
import configuration
import lewansoul_wrapper

# Python 2 does not have a constant for infinity. (Python 3 added math.inf.)
infinity = float("inf")

class roverwheel:
  """
  Rover wheel class tracks information specific to a particular wheel on
  the chassis.
  """
  def __init__(self, name, x=0, y=0, rollingcontrol=None,rollingparam=None,
    steeringcontrol=None, steeringparam=None):

    # String is used to identify this wheel in various operations. Bonus if
    # the words make sense to a human reader ('front_left') but not required.
    self.name = name

    # X,Y coordinate of this wheel on the rover chassis, relative to center.
    self.x = x
    self.y = y

    # Rolling velocity motor (optional): a reference to the control object and
    # the parameters to identify this wheel to the control.
    self.rollingcontrol = rollingcontrol
    self.rollingparam = rollingparam

    # Steering angle motor (optiona): similar to the above, but for steering.
    self.steeringcontrol = steeringcontrol
    self.steeringparam = steeringparam

    # The most recently commanded steering angle and rolling velocity
    self.angle = 0
    self.velocity = 0

    # If we were given a rolling velocity control, run any initialization we
    # need and obtain its label string to show to user.
```

```
    if self.rollingcontrol:
      self.rollingcontrol.init_velocity(self.rollingparam)
      try:
        self.rollinglabel = self.rollingcontrol.version(self.rollingparam)
      except ValueError as ve:
        self.rollinglabel = "(No Response)"

    # Repeat the above, this time for steering angle control.
    if self.steeringcontrol:
      self.steeringcontrol.init_angle(self.steeringparam)
      try:
        self.steeringlabel = self.steeringcontrol.version(self.steeringparam)
      except ValueError as ve:
        self.steeringlabel = "(No Response)"

  def poweroff(self):
    """
    Instructs the motor controller to stop rolling, stop holding position,
    whatever is the least-effort situation. (If applicable)
    """
    self.velocity = 0
    if self.rollingcontrol:
      self.rollingcontrol.power_percent(self.rollingparam, 0)

    # Killing the power leaves the angle wherever it was last (except as
    # moved by external forces) so leave self.angle alone.
    if self.steeringcontrol:
      self.steeringcontrol.power_percent(self.steeringparam, 0)

  def anglevelocity(self):
    """
    Send the dictated angle and velocity to their respective controls
    """
    if self.rollingcontrol:
      self.rollingcontrol.velocity(self.rollingparam, self.velocity)

    if self.steeringcontrol:
      self.steeringcontrol.angle(self.steeringparam, self.angle)

  def steerto(self, angle):
    """
    Steer this wheel to the specified angle. Caller is responsible for
    validation of all parameters.
    """
    self.steeringcontrol.angle(self.steeringparam, angle)

  def steersetzero(self):
    """
    Set the current steering angle of this wheel as the new zero. Caller is
    responsible for validation of all parameters
    """
    self.steeringcontrol.steer_setzero(self.steeringparam)

  def motor_voltage(self):
    """
    Query the rolling and steering motor controllers for their current input
    voltage levels
```

```
    """
    voltages = dict()

    if self.rollingcontrol:
      voltages["Rolling"] = self.rollingcontrol.input_voltage(self.rollingparam)
    else:
      voltages["Rolling"] = "Not Applicable"

    if self.steeringcontrol:
      voltages["Steering"] = self.steeringcontrol.input_voltage(self.
    steeringparam)
    else:
      voltages["Steering"] = "Not Applicable"

    return voltages

class chassis:
  """
  Rover chassis class tracks the physical geometry of the chassis and uses
  that informaton to calculate Ackerman steering angles and relative
  velocity for wheel travel
  """

  def __init__(self):
    # List of wheels
    # Each wheel is a dictionary mapping name of a wheel to its specific info.
    self.wheels = dict()

    # When turning radius grows beyond this point, the wheel angles are so
    #   miniscule it is indistinguishable from straight line travel. This hard
    #   coded default can be updated based on chassis config by calling
    #   calculate_radius_min_max()
    self.maxRadius = 250

    # Radius representing the tightest turn this chassis can make. Minimum
    #   value of zero indicates chassis is capable of turning in place.
    #   This hard coded default value can be updated based on chassis config by
    #   calling calculate_radius_min_max()
    self.minRadius = 17.75

    # The current (velocity, radius) that dictated wheel angle and velocity.
    #   Velocity unit is up to the caller, math works regardless of units
    #     inches, metric, quadrature pulses, etc.
    #   Radius unit must match those used to specify wheel coordinates.
    self.currentMotion = (0, infinity)

    # A dictionary mapping a name string identifying a motor controller type
    #   to an instance of the motor controller.
    self.motorcontrollers = dict()

  def init_motorcontrollers(self):
    """
    Creates the dictionary where a name in the configuration file can be
    matched with its corresponding motor controller.
    """
    try:
      lws = lewansoul_wrapper.lewansoul_wrapper()
```

```
      lws.connect()
      self.motorcontrollers['lewansoul'] = lws
      #print "------------------Funciona be fins aqui----------------------"
    except StandardError as se:
      logging.getLogger(__name__).error("Unable to initialize LewanSoul Servo
    Library: %s",str(se))

  def ensureready(self):
    """
    Makes sure this chassis class is ready for work by ensuring the required
    information is loaded and ready.
    """
    if len(self.wheels) > 0:
      return

    # Initialize motor controller dictionary.
    self.init_motorcontrollers()

    # Load configuration from JSON.
    config = configuration.configuration("roverchassis")
    wheeljson = config.load()

    # Using the data in configuration JSON file, create a wheel object.
    for wheel in wheeljson:
      # Retrieve name and verify uniqueness.
      name = wheel['name']
      if name in self.wheels:
        raise ValueError("Duplicate wheel name {} encountered.".format(name))

      # Initialize all optional motor control values to None
      steeringcontrol = None
      steeringparam = None
      rollingcontrol = None
      rollingparam = None

      # Fill in any rolling velocity motor control and associated parameters
      rolling = wheel['rolling']
      if rolling:
        rollingtype = rolling[0]
        if len(rolling) == 2:
          rollingparam = rolling[1]
        else:
          #print "Entro aqui " * 5
          rollingparam = rolling[1:]
          #print "--------------"
          #print rollingparam
          #print rollingtype
          #print self.motorcontrollers
          #print "--------------"
        if rollingtype in self.motorcontrollers:
          rollingcontrol = self.motorcontrollers[rollingtype]
        else:
          #print "---------"
          #print rolling
          #print "---------"
          raise ValueError("Not working motor")
```

```
      # Fill in any steering angle motor control and associated parameters
      steering = wheel['steering']
      if steering:
        steeringtype = steering[0]
        if len(steering) == 2:
          #print "---------------aqui--------------"
          steeringparam = steering[1]
        else:

          steeringparam = steering[1:]
        if steeringtype in self.motorcontrollers:
          steeringcontrol = self.motorcontrollers[steeringtype]
        else:
          raise ValueError("Unknown motor control type")

      # Add the newly created roverwheel object to wheels dictionary.
      self.wheels[name] = roverwheel(name, wheel['x'], wheel['y'],
        rollingcontrol, rollingparam, steeringcontrol, steeringparam)

    # Update radius min/max based on the rover chassis configuration info
    self.calculate_radius_min_max()

    # Wheels are initialized, set everything to zero.
    self.move_velocity_radius(0)

  def move_velocity_radius(self, velocity, radius=infinity):
    """
    Given the desired velocity and turning radius, update the angle and
    velocity required for each wheel to perform the desired motion.

    Velocity and radius is given relative to rover center.

    Radius of zero indicates a turn-in-place movement. (Not yet implemented)
    Radius of infinity indicates movement in a straight line.
    """
    if abs(radius) < self.minRadius:
      # This chassis configuration could not make that tight of a turn.
      raise ValueError("Radius below minimum")

    if abs(velocity) > 100:
      raise ValueError("Velocity percentage may not exceed 100")

    self.currentMotion = (velocity, radius)

    if radius > self.maxRadius:
      # Straight line travel
      for wheel in self.wheels.values():
        wheel.angle = 0
        wheel.velocity = velocity
    else:
      # Calculate angle and velocity for each wheel
      for wheel in self.wheels.values():
        # Dimensions of triangle representing the wheel. Used for calculations
        # in form of opposite, adjacent, and hypotenuse
        opp = wheel.y
        adj = radius-wheel.x
        hyp = math.sqrt(pow(opp,2) + pow(adj,2))
```

```
        # Calculate wheel steering angle to execute the commanded motion.
        if adj == 0:
          wheel.angle = 90
        else:
          wheel.angle = math.degrees(math.atan(float(opp)/float(adj)))

        # Calculate wheel rolling velocity to execute the commanded motion.
        if radius == 0:
          wheel.velocity = 0 # TODO: Velocity calculation for spin-in-place
    where radius is zero
        else:
          wheel.velocity = velocity * hyp/abs(radius)

        # If center of rotation is within the wheel track, and between the
        # wheel and the origin, then this wheel will need to turn in the
        # opposite direction so the rover body can turn about the center.
        if (radius < 0 and wheel.x < 0 and wheel.x < radius) or (radius > 0 and
    wheel.x > 0 and wheel.x > radius):
          wheel.velocity = -wheel.velocity

    # Go back and normalize al the wheel roll rate magnitude so they are at or
    # below target velocity while maintaining relative ratios between their
    rates.
    maxCalculated = 0

    for wheel in self.wheels.values():
      if abs(wheel.velocity) > maxCalculated:
        maxCalculated = abs(wheel.velocity)

    if maxCalculated > velocity:
      # At least one wheel exceeded specified maxVelocity, calculate
      # normalization ratio and apply to every wheel.
      reductionRatio = abs(velocity)/float(maxCalculated)
      for wheel in self.wheels.values():
        wheel.velocity = wheel.velocity * reductionRatio

    # We're sending commands for a particular wheel - steering and rolling
    # velocity - before we move on to the next wheel. If this causes timing
    # issues (wheels start moving before they've finished pointing in the
    # right direction, etc.) we may have to send all steering commands first,
    # wait until we reach the angles, before sending velocity commands.
    for wheel in self.wheels.values():
      wheel.anglevelocity()

  def calculate_radius_min_max(self):
    """
    Once the wheel configuraton has been loaded, read maximum turning ability
    of the wheels and calculate the minimum turning radius. Also look at the
    radius when the wheels are turned a single degree and use that as maximum
    turning radius.

    Initial values are established by first finding the wheel with the
    greatest X distance from center. Minimum radius is 1% of its distance, and
    maximum is 10 times (1000%) of the distance.
    """
    wheel_x_max = 0
```

```
    for wheel in self.wheels.values():
      if abs(wheel.x) > wheel_x_max:
        wheel_x_max = abs(wheel.x)

    if wheel_x_max > 0:
      limit_min = wheel_x_max * 0.01
      limit_max = wheel_x_max * 10

    # Initial values established, now let's look at each wheel and calculate
    # its relative min/max and compare against initial values.
    for wheel in self.wheels.values():
      if wheel.steeringcontrol:
        angle_max = wheel.steeringcontrol.maxangle(wheel.steeringparam)
        if angle_max < 90:
          limit_radius = wheel.x + (wheel.y/math.tan(math.radians(angle_max)))
          if limit_radius > limit_min:
            limit_min = limit_radius
          # If the rover uses steering mechanism that can be more precise than
          # +/- one degree, decrease the value accordingly.
          limit_radius = wheel.x + (wheel.y/math.tan(math.radians(1)))
          if abs(limit_radius) < limit_max:
            limit_max = abs(limit_radius)

    self.minRadius = limit_min
    self.maxRadius = limit_max
```

Mòdul 7: roverchassis.py

```
"""
MIT License

Copyright (c) 2018 Roger Cheng

"""
import roverchassis

class JuliusRover():
    def __init__(self):
        """
        Init make a roverchassis class
        """
        self.chassis = roverchassis.chassis()
        self.rover_ready()
        print ("Rover ready to go")

    def rover_ready(self):
        """
        Rover preparat
        """
        self.chassis.ensureready()

    def stop_motors(self):
        """
        Stop motors immediately
        """
        self.rover_ready()
        for wheel in self.chassis.wheels.values():
          wheel.poweroff()
```

```
        print("Motors stopped")

    def drive(self, magnitude, angle):
        """
        Allows user to send a single angle+velocity command to chassis.
        """
        self.rover_ready()
        if angle == 0:
            radius = float("inf")
        elif angle>0:
            radius = self.chassis.minRadius + (self.chassis.maxRadius - self.
    chassis.minRadius) * (100 - angle) / 100.0
        else:
            radius = -self.chassis.minRadius - (self.chassis.maxRadius - self.
    chassis.minRadius) * (100 + angle) / 100.0

        self.chassis.move_velocity_radius(magnitude, radius)
        print "Moving with magnitude " + str(magnitude) + " and radius " + str(
    radius)

if __name__ == '__main__':
    rover = JuliusRover()
    rover.drive(30, 40)
    rover.stop_motors()
```

Mòdul 8: menu.py

```
import time
import os
import sys
import subprocess
import errno

FIFO = 'mypipe'

print "Creem servidors de Julius"
#Creem el servidor amb un subprocess
try:
    process_servidor = subprocess.Popen("padsp julius -input mic -C /home/pi/
    Desktop/Sample.jconf -module", shell = True, stdout = subprocess.PIPE)
except:
    raise ValueError("No funciona el servidor")

print "Servidor creat"

try:
    os.mkfifo(FIFO)
except OSError as oe:
    if oe.errno != errno.EEXIST:
        print "LA FIFO ja existeix"
        pass

print "Creant pipe si no existeix"
try:
    fifo = open(FIFO, 'w')
except:
    pass
```

```
fifo.close()
print "Pipe preparada"

while True:
    output_servidor = str(process_servidor.stdout.readline())
    print output_servidor
    if str(output_servidor[0:21]) == "Error: adin_alsa: una":
        fifo = open(FIFO,'w')
        fifo.write("Micro out")
        fifo.close()
        time.sleep(1)#Per donar temps al gestor a executar accions

    if str(output_servidor[0:22]) == "Warning: strip: sample" :
        fifo = open(FIFO,'w')
        fifo.write("Microfon no funciona")
        fifo.close()
        time.sleep(1)#Per donar temps al gestio a executar accions
```

Mòdul 9: make_server.py

```
"""
MIT License

Copyright (c) 2018 Roger Cheng

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""
import serial
import serial.tools.list_ports
import time
from struct import *

import configuration

def bytetohex(bytearray):
  """
  Returns hexadecimal string representation of byte array
  Copied from StackOverflow
  https://stackoverflow.com/questions/19210414/byte-array-to-hex-string
  """
  return ''.join('{:02x}'.format(x) for x in bytearray)
```

```
class lewansoul_wrapper:
  """
  Class that implements the rover motor control methods for serial bus
  servo by LewanSoul. Specifically their model LX-16A.
  """
  def __init__(self):
    self.sp = None
    self.port_change_ok = True

  def check_sp(self):
    """ Raises error if we haven't opened serial port yet. """
    if self.sp == None:
      raise ValueError("LewanSoul serial communication is not available.")

  def connect(self):
    """
    Read serial port connection parameters from JSON configuration file
    and open the port.
    """

    # Read parameter file
    config = configuration.configuration("lewansoul")
    connectparams = config.load()['connect']

    # Open serial port with parameters
    s = serial.Serial()
    s.baudrate = connectparams['baudrate']
    s.port = connectparams['port']#Deixem per tenir un per defecte en el cas que
     no hi hagi cap conectat
    self.change_device(s)
    print "***************"
    print "Port that has changed"
    print s.port
    print "***************"
    s.timeout = connectparams['timeout']
    s.open()

    if s.is_open:
      self.sp = s

  def change_device(self, s):
    """
    Function to change device
    """
    ports_instance = serial.tools.list_ports.comports()
    ports_available = list()
    for ports in ports_instance: #vid no canvia fer comprovacio amb id
      if str(ports.vid) == "6790":
        s.port = str(ports.device)
        self.port_change_ok = True
        break

  def close(self):
    """
    Closes down the serial port
    """
    if self.sp.is_open:
```

```
      self.sp.close()
      self.sp = None

  def send(self, servo_id, command, data=None):
    """
    Send a command to a LewanSoul servo, taking care of the header and
    checksum calculation for a command packet.
    """
    self.check_sp()
    packet = [0x55, 0x55]

    if servo_id < 0 or servo_id > 0xfe:
      raise ValueError("Servo ID {} is out of valid range".format(servo_id))
    packet.append(servo_id)

    length = 3
    if data:
      length = length + len(data)
    #TODO: check maximum length
    packet.append(length)

    #TODO: Check for valid command
    packet.append(command)

    if data:
      for d in data:
        packet.append(d)

    checksum = servo_id + length + command
    if data:
      for d in data:
        checksum = checksum + d
    checksum = (~checksum) & 0xff
    packet.append(checksum)

    packet_bytes = bytearray(packet)
    # print("Sending command byte stream of {}".format(bytetohex(packet_bytes)))
    try:
      self.sp.write(packet_bytes)
    except:
      conection_validated = False
      self.port_change_ok = False
      while(self.port_change_ok == False):
  while(conection_validated == False):
          try:
            self.connect()
            print("In the loop")
            conection_validated = True
          except:
            conection_validated = False
            time.sleep(1)
            print "Not possible to make a serial comunication"
      self.sp.write(packet_bytes)
      self.port_change_ok = True
      print("Port changed correctly")

  def read_raw(self, length=100):
```

```
    """
    Reads a stream of bytes from serial device and returns it without any
    attempts at parsing or validation
    """
    self.check_sp()
    return bytearray(self.sp.read(length))

  def read_parsed(self, length=100, expectedid=None, expectedcmd=None,
    expectedparams=None):
    """
    Reads up to 'length' bytes and parse it according to pack format spec
    from "LewanSoul Bus servo Communication Protocol" PDF:

    0      1      2      3      4      [ ... ]
    0x55   0x55   ID     Len    Cmd    Param1... ParamN   Checksum

    Length of data is all bytes after ID, including length byte.
    Packet with no parameters has length of 3 bytes, plus header+ID = 6 bytes
    total.
    Checksum = (~(ID+Length+Cmd+Param1+...+ParamN)) & 0xFF

    - -

    Optional parameters:
      expectedid = if provided, will check message ID against expected ID.
      expectedcmd = if provided, will check message command against expected
    command.
      expectedparams = if provided, will check the number of bytes in parameter
    matches expected.

      If a mismatch is found, a ValueError is raised.
    """
    self.check_sp()
    r = bytearray(self.sp.read(length))
    print len(r)

    # Check response length
    if len(r) < 6:
      raise ValueError("Need at least 6 bytes for a valid packet, received {}".
    format(len(r)))

    # Check header
    if r[0] != 0x55 or r[1] != 0x55:
      raise ValueError("Response header is {:02x} {:02x}, expected 0x55 0x55".
    format(r[0], r[1]))

    # Verify length
    rlen = r[3]
    if rlen+3 != len(r):
      raise ValueError("Packet claims to have {} bytes of data + 3 bytes of
    header, but we retrieved {} bytes.".format(rlen, len(r)))

    # Verify checksum
    checksum = 0
    for b in r[2:-1]:
      checksum = checksum + b
    checksum = (~checksum) & 0xFF
```

```
    if checksum != r[-1]:
      raise ValueError("Packet checksum {} does not match calculated checksum {}
    ".format(r[-1], checksum))

    # If an expected ID is given, compare against ID in the message.
    rid = r[2]
    if expectedid != None and expectedid != rid:
      raise ValueError("Response stamped with ID {}, expected {}".format(rid,
    expectedid))

    # If an expected command is given, compare against command in the message.
    rcmd = r[4]
    if expectedcmd != None and expectedcmd != rcmd:
      raise ValueError("Response command {}, expected {}".format(rcmd,
    expectedcmd))

    # Examine parameters, if any.
    if rlen > 3:
      rparams = bytearray(r[5:-1])
      if expectedparams != None and expectedparams != len(rparams):
        raise ValueError("Received {} bytes of parameters, expected {}".format(
    len(rparams),expectedparams))
    else:
      rparams = None
      if expectedparams != None and expectedparams != 0:
        raise ValueError("Received {} bytes of parameters, expected none".format
    (len(rparams)))

    # Return results in a tuple
    return (rid, rcmd, rparams)

  def version(self, id):
    """ Identifier string for this motor controller """
    return "LewanSoul"

  @staticmethod
  def check_id(id):
    """ Verifies servo ID is within range and inverted status is boolean"""
    if not isinstance(id, (tuple,list)):
      raise ValueError("LewanSoul identifier must be a tuple")

    if not isinstance(id[0], int):
      raise ValueError("LewanSoul servo address must be an integer")

    if id[0] < 0 or id[0] > 253:
      raise ValueError("LewanSoul servo address {} outside of valid range 0-253"
    .format(id[0]))

    if not isinstance(id[1], int):
      raise ValueError("LewanSoul servo center position must be an integer")

    if not isinstance(id[2], bool):
      raise ValueError("Inverted status must be a boolean")

    return tuple(id)
```

```
  def power_percent(self, id, percentage):
    """ Runs servo in motor mode at specified +/- percentage """
    sid, center, inverted = self.check_id(id)
    self.check_sp()

    pct = int(percentage)
    if abs(pct) > 100:
      raise ValueError("Motor power percentage {0} outside valid range from 0 to
     100.".format(pct))

    # LewanSoul API wants power expressed between -1000 and 1000, so multiply by
     10.
    power = percentage*10

    if inverted:
      power = power * -1

    self.send(sid, 29, bytearray(pack('hh',1,power)))

  def set_max_current(self, id, current):
    """ LewanSoul does not support overpower protection. """
    sid, center, inverted = self.check_id(id)
    self.check_sp()
    # Does nothing

  def init_velocity(self, id):
    """ Sets LewanSoul into motor mode and speed zero """
    sid, center, inverted = self.check_id(id)
    self.check_sp()

    self.send(sid, 29, bytearray(pack('hh',1,0)))

  def velocity(self,id,pct_velocity):
    """
    Runs the specified servo in motor mode at specified velocity
    In case of LewanSoul servos, it is the same as power_percent.
    """
    self.power_percent(id,pct_velocity)

  def init_angle(self, id):
    """
    Sets the LewanSoul into servo mode and move to center over 2 seconds
    """
    sid, center, inverted = self.check_id(id)
    self.check_sp()

    self.send(sid, 29, (0,0,0,0)) # Servo mode
    self.send(sid, 1, bytearray(pack('hh', center, 2000)))

  def maxangle(self, id):
    sid, center, inverted = self.check_id(id)
    self.check_sp()
    return 120

  def angle(self, id, angle):
    sid, center, inverted = self.check_id(id)
    self.check_sp()
```

```
    if abs(angle) > 95:
      raise ValueError("Steering angle {} exceeded expected maximum of 90".
    format(angle))

    delta = angle * (500.0/120.0) # 500 count/ 120 degrees = counts per degree.

    if inverted:
      delta = delta * -1

    self.send(sid, 29, (0,0,0,0)) # Servo mode
    self.send(sid, 1, bytearray(pack('hh', center+delta, 200)))

  def steer_setzero(self, id):
    sid, center, inverted = self.check_id(id)
    self.check_sp()
    # TODO: Support live adjustment

  def input_voltage(self, id):
    """
    Query LewanSoul servo's internal voltage monitor
    """
    sid, center, inverted = self.check_id(id)
    self.check_sp()

    self.send(sid, 27)
    (sid, cmd, params) = self.read_parsed(length=8, expectedcmd=27,
    expectedparams=2)
    millivolts = unpack('h', params)[0]

    return millivolts/1000.0

if __name__ == "__main__":
  """
  Command line interface to work with LewanSoul serial bus servos.
  Implements a subset of the servo's functionality
  * Move to position over time. (Servo mode)
  * Spin at a specified speed. (Motor mode)
  * Broadcast query for servo ID
  * Rename servo to another ID
  * Unload and power down motors
  """
  import argparse

  parser = argparse.ArgumentParser(description="LewanSoul Serial Servo Command
    Line Utility")

  parser.add_argument("-id", "--id", help="Servo identifier integer 0-253. 254
    is broadcast ID.", type=int, default=1)
  parser.add_argument("-t", "--time", help="Time duration for action", type=int,
     default=0)
  group = parser.add_mutually_exclusive_group()
  group.add_argument("-m", "--move", help="Move servo to specified position
    0-1000", type=int)
  group.add_argument("-q", "--queryid", help="Query for servo ID", action="
    store_true")
  group.add_argument("-r", "--rename", help="Rename servo identifier", type=int)
```

```
group.add_argument("-s", "--spin", help="Spin the motor at a specified speed
  from -1000 to 1000", type=int)
group.add_argument("-u", "--unload", help="Power down servo motor", action="
  store_true")
group.add_argument("-v", "--voltage", help="Read current input voltage",
  action="store_true")
args = parser.parse_args()

c = lewansoul_wrapper()
c.connect()

if args.move != None: # Explicit check against None because zero is a valid
  value
  if args.move < 0 or args.move > 1000:
    print("Servo move destination {} is outside valid range of 0 to 1000 (1000
   = 240 degrees)".format(args.move))
  elif args.time < 0 or args.time > 30000:
    print("Servo move time duration {} is outside valid range of 0 to 30000
  milliseconds".format(args.time))
  else:
    print("Moving servo {} to position {}".format(args.id, args.move))
    c.send(args.id, 29, (0,0,0,0)) # Turn on servo mode (in case it was
  previously in motor mode)
    c.send(args.id, 1, bytearray(pack('hh', args.move, args.time)))
elif args.queryid:
  print("Broadcasting servo ID query")
  c.send(0xfe, 14) # Broadcast and ask to report ID
  (sid, cmd, params) = c.read_parsed(length=7, expectedcmd=14, expectedparams
  =1)
  if sid != params[0]:
    raise ValueError("ID response stamped with {} but payload says {}".format(
  sid, params[0]))
  print("Servo ID {} responded to query".format(sid))
elif args.rename:
  print("Checking the specified servo ID {} is on the serial network.".format(
  args.id))
  c.send(args.id, 14) # Ask for current servo ID
  (sid, cmd, params) = c.read_parsed(length=7, expectedcmd=14, expectedparams
  =1)
  if sid != args.id or params[0] != args.id:
    print("Unexpected answer from servo {} when verifying servo {} is on the
  network.".format(sid, args.id))
  else:
    print("Checking the specified destination servo ID {} is not already taken
  .".format(args.rename))
    c.send(args.rename, 14)
    expectempty=c.read_raw()
    if len(expectempty) > 0:
      raise ValueError("Someone answers to servo ID {} on the network, rename
  aborted.".format(args.rename))
    else:
      print("Renaming servo ID {} to {}".format(args.id, args.rename))
      c.send(args.id, 13, (args.rename,))
      print("Verifying the servo now answers to new ID")
      c.send(args.rename, 14)
      (sid, cmd, params) = c.read_parsed(length=7, expectedcmd=14,
  expectedparams=1)
```

```
        if sid != args.rename or sid != params[0]:
          print("Querying for response from ID {} failed, we got answer from ID
    {}/{} instead.".format(args.rename, sid, params[0]))
        else:
          print("Servo successfully renamed to ID {}".format(args.rename))
  elif args.spin != None: # Zero is a valid parameter.
    if args.spin < -1000 or args.spin > 1000:
      print("Servo spin speed {} is outside valid range of -1000 to 1000".format
    (args.spin))
    else:
      print("Spinning motor of servo {} at rate of {}".format(args.id, args.spin
    ))
      c.send(args.id, 29, bytearray(pack('hh', 1, args.spin)))
  elif args.unload:
    c.send(args.id, 31, (0,))
  elif args.voltage:
    c.send(args.id, 27)
    (sid, cmd, params) = c.read_parsed(length=8, expectedcmd=27, expectedparams
    =2)
    voltage = unpack('h', params)[0]
    print("Servo {} reports input voltage of {}".format(sid, voltage/1000.0))
  else:
    # None of the actions were specified? Show help screen.
    parser.print_help()

  c.close()
```

Mòdul 10: lewansoul_wrapper.py

```
#!/usr/bin/env python
import time
import sys
import pyjulius
import Queue
import os
import errno

FIFO = 'mypipe'

# Initialize and try to connect
client = pyjulius.Client('localhost', 10500)
try:
    client.connect()
except pyjulius.ConnectionError:
    print 'Start julius as module first!'
    sys.exit(1)

#Es crea la fifo si no existeix
try:
    os.mkfifo(FIFO)
except OSError as oe:
    if oe.errno != errno.EEXIST:
        print "LA FIFO ja existeix"
        pass

# Start listening to the server
client.start()
```

```
try:
    while 1:
        try:
            result = client.results.get(False)
            print result
        except Queue.Empty:
            print "Impossible rebre res"
            continue
        if isinstance(result, pyjulius.Sentence):
            fifo = open(FIFO,'w')
            fifo.write(str(result))
            fifo.close()
        else:
            print "S'ha rebut algo incorrecte"

except KeyboardInterrupt:
    print 'Exiting...'
    client.stop()  # send the stop signal
    client.join()  # wait for the thread to die
    client.disconnect()  # disconnect from julius
```

Mòdul 11: connect_to_server.ppy

```
#!/usr/bin/env python
import os
import sys
import pyjulius
import os
import errno
import time
import subprocess
from multiprocessing import Queue
from threading import Thread
from menu import JuliusRover

if __name__ == '__main__':
    pipe = 'mypipe'
    #Es crea la pipe si no existeix
    try:
        os.mkfifo(pipe)
    except OSError as fifo:
        pass
    rover = JuliusRover()
    rover.rover_ready()
    velocity_rover = 20
    angle_rover = 0
    while True:
        with open(pipe) as fifo:
            print "Fifo opened"
            while True:
                try:
                    data = fifo.read()
                    print data
                    if str(data) == "Microfon no funciona":
                        print "Microfon no funciona"
                        rover.stop_motors()

                    if str(data) == "Warning: strip: sample" :
```

```
                print "Microfon no funciona"
                rover.stop_motors()

            if str(data) == "rover stop":
                print "I stop"
                rover.stop_motors()

            if str(data) == "rover up":
                if velocity_rover < 0:
                    velocity_rover = -velocity_rover
                    rover.drive(velocity_rover, angle_rover)

            if str(data) == "rover down":
                if velocity_rover > 0:
                    velocity_rover = -velocity_rover
                    rover.drive(velocity_rover, angle_rover)

            if str(data) == "rover fast":
                if velocity_rover == 100:
                    print "I can't go faster"
                else:
                    velocity_rover += 10
                rover.drive(velocity_rover, angle_rover)

            if str(data) == "rover slow":
                if velocity_rover == 10:
                    print "I can't go slower"
                else:
                    velocity_rover -= 10
                rover.drive(velocity_rover, angle_rover)

            if str(data) == "rover straight":
                angle_rover = 0
                rover.drive(velocity_rover, angle_rover)

            if str(data) == "rover right":
                if angle_rover < 0:
                    angle_rover = 0
                if angle_rover != 100:
                    angle_rover += 10
                else:
                    print "I can't turn more"
                rover.drive(velocity_rover, angle_rover)
        except:
            print "Algo ha fallat"
```

Mòdul 12: main.py